A PRIMERA REVISTA DE PROGRAMACION EN CASTELLANO

SOLO PROGRAMADORES

O VI. NÚMERO 56

UNA PUBLICACIÓN DE: TOWER

975 Ptas. • 5,86 € (IVA incluido)

EL NUEVO ESTÁNDAR EN INTERNET

XML





BASES DE DATOS
Acceso a Oracle 8 desde C++ Builder

VISUAL BASIC
Aplicaciones IIS

PROGRAMACIÓN MULTIMEDIA
Multimedia con Java: sonido y vídeo

INTRANET/INTERNET
Del HTML al acceso a bases de datos

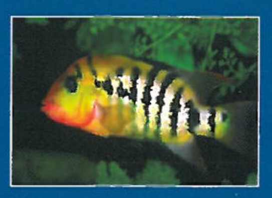
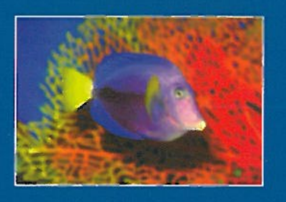
COMUNICACIONES
Desarrollo de aplicaciones con videoconferencia

PROGRAMACIÓN INTERNET
Cómo crear un buscador Web

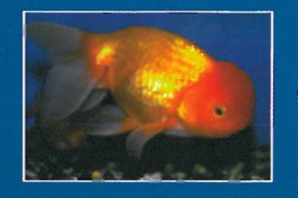
NUEVAS TECNOLOGÍAS
DirectX 6.1

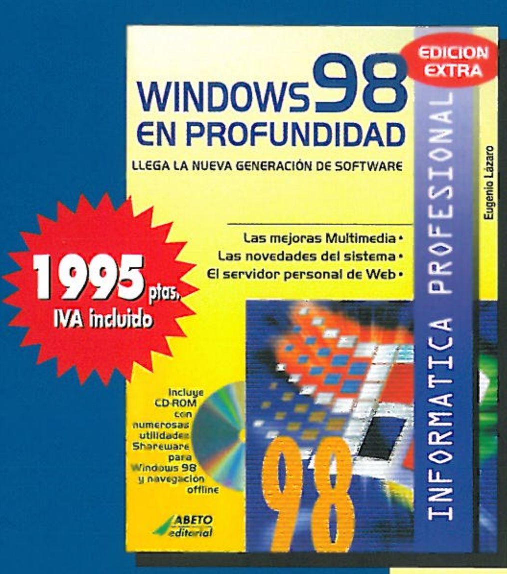
CONTENIDO DEL CD-ROM

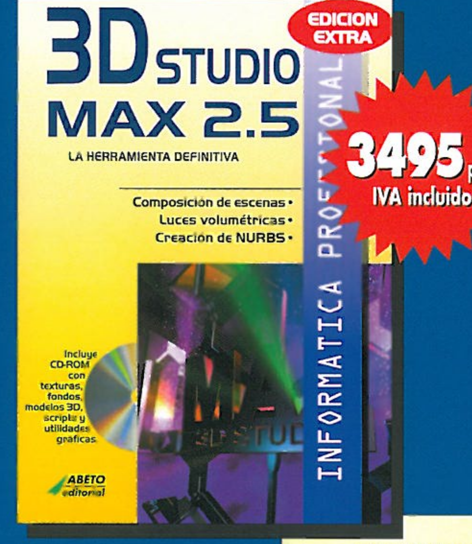
DIRECTX 6.1 • SQL SERVER 7.0 • PERL DEVELOPMENT KIT 1.1 • FIREWORKS 2.0 • VISUAL ROUTE 4.0c

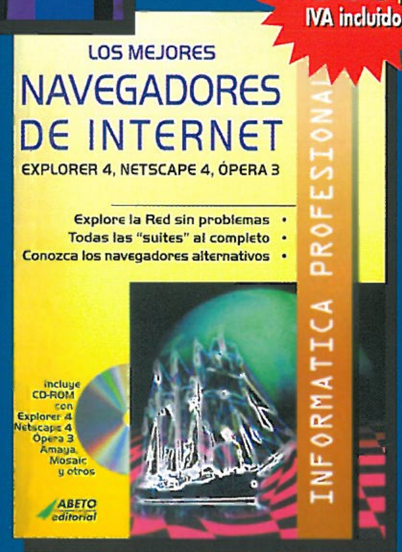
PARA  MOVERTE  EN 

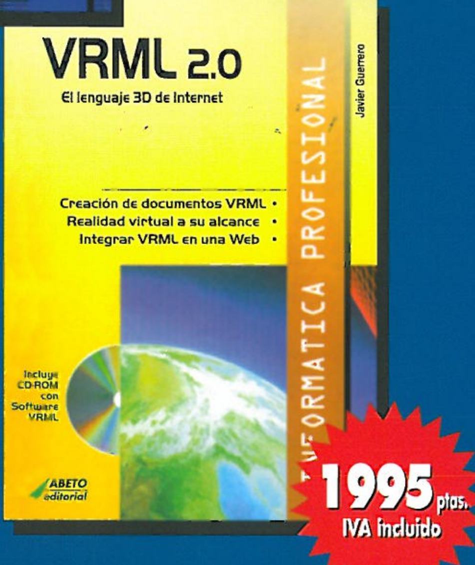
INFORMÁTICA  COMO  PEZ

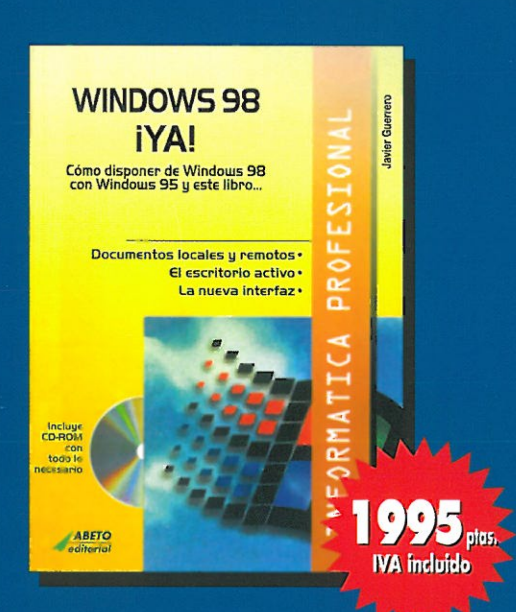
 EN  EL  AGUA 











Incluyen CD-ROM de regalo

Ya a la venta en quioscos y librerías.

ABETO editorial

c/ Aragoneses, 7 • 28108 Alcobendas (Madrid)
Tel.: 91 661 42 11* • Fax: 91 661 43 86

SÓLO PROGRAMADORES

Número 56
SÓLO PROGRAMADORES
es una publicación de
TOWER COMMUNICATIONS
Director Editor
Antonio M. Ferrer Abelló
aferrer@towercom.es
Subdirector
Oscar Rodríguez Fernández
oscarrf@towercom.es
Coordinador Técnico
Eduardo De Riquer Frutos
eriquer@towercom.es
Coordinadora de Redacción
Erika de la Riva Arnáiz
eriva@towercom.es
Colaboradores
Constantino Sánchez, Juan M. Menéndez, Jorge Delgado, Javier Sanz, Adolfo Aladro, Enrique de la Lastra, Rafael Corchuelo, Victoria Rus, Esteban Amado, Jordi Agost
Maquetación y Tratamiento de Imagen
Ana Isabel Madero Bocos

Publicidad
Inmaculada Romera (Madrid)
Tel.: (91) 661 42 11
Pepín Gallardo (Barcelona)
Tel.: (93) 213 42 29

Suscripciones
Alicia Zazo
Tel. (91) 661 42 11 Fax: (91) 661 43 86
suscrip@towercom.es

Laboratorio
Javier Amado (Jefe)
jamado@towercom.es
Servicio Técnico
Manuel Hernando
mhernando@towercom.es

Preimpresión
IndesColor
Impresión
Gráficas Muriel
Distribución
SGEL
Distribución en Argentina / Chile / Colombia / México / Venezuela
Capital: Huesca y Sanabria
Interior: D.G.P.

TOWER COMMUNICATIONS
Director General
Antonio M. Ferrer Abelló
Director Financiero
Francisco García Díaz de Liaño
Director de Producción
Carlos Peropadre
Directora Comercial
Carmina Ferrer
carmina@towercom.es
Distribución
Almiro Sanguino

Redacción, Publicidad y Administración
C/ Aragoneses, 7
28108 Pol. Ind. Alcobendas (MADRID)
Telf.: (91) 661 42 11 / Fax: (91) 661 43 86

Depósito legal: M-26827-1994
ISSN: 1134-4792
PRINTED IN SPAIN



Lo que ves es lo que no quieres ver

¿Alguien se ha enfrentado con la horrible necesidad de publicar o crear un documento en *HTML*, ya sea por gusto, trabajo o devoción? La mayoría pensará que la pregunta huelga en una revista donde gran parte de los artículos se encuentran relacionados con *Internet* y donde el nivel de los lectores hace sospechar que el adjetivo horrible no está muy relacionado con la programación. Y así es, ya que a la hora de crear una página de estas características lo de menos es el análisis del problema, el diseño, la presentación, el trato con el cliente y demás "asuntillos" sin interés. Alguien pensará que unas vacaciones me sentarán como una bendición, pero aún en ese esperado supuesto, el gran y espantoso problema surge cuando utilizamos alguna de las maravillosas herramientas *WYSIWYG*. Esto puede parecer extraño si tenemos en cuenta que se caracterizan por su facilidad de manejo y por la posibilidad de realizar diseños de forma visual.

La verdad es que hasta aquí todo son parabienes, pero ¿qué ocurre cuando abrimos esa misma página con otra herramienta? ¿lo han adivinado? Efectivamente, lo más probable es que el parecido con el documento diseñado sea una ficción, puesto que cada herramienta sigue los estándares en función de criterios empresariales, añadiendo elementos propietarios (e incompatibles con los ofrecidos por otras empresas, claro). Entonces me pregunto, para qué sirve la palabra "estándar" si cada empresa puntera sigue el suyo propio y las herramientas de diseño vuelven locos a los que las utilizan. Para colmo, los males no terminan aquí, ya que las páginas se transmiten y se encuentran disponibles en todos los ordenadores del mundo gracias a *Internet*. (Miedo me da escribir estas afirmaciones por si algún periodista de sucesos publica que el mal se transmite como una plaga por todo el planeta debido a la maligna red *Internet*, pero éste es otro tema). Y es que lo peor de todo ocurre cuando visualizamos la página con diferentes navegadores, de tal forma que nuestra magnífica obra resulta un insulto para la vista en otro distinto y es necesario desarrollar páginas diferentes para todos los navegadores, sin olvidar cada una de las versiones existentes.

Supongo que ahora podremos estar o no de acuerdo, pero espero que el planteamiento inicial quede algo más claro para todos. Es muy probable que si alguna vez existe un Renacimiento Informático, los grandes genios de la época utilicen un editor de textos cualquiera e introduzcan directamente las etiquetas del lenguaje *HTML*. Por lo menos sabrán a qué atenerse y no se sentirán engañados después de terminar su trabajo. También cabe la posibilidad de que la palabra estándar deje de ser un absurdo eufemismo y las grandes empresas se pongan de acuerdo para crear uno auténtico, aunque desde luego no resulta aconsejable prescindir de programas como *edit*, *worpad* ó *vi*, por si acaso.

SÓLO PROGRAMADORES



INPRISE LANZA AL MERCADO LA NUEVA VERSIÓN DE BORLAND C++ BUILDER 4

El pasado 2 de Febrero de 1.999, *Borland*, la división de la corporación *Inprise* encargada de la creación de herramientas de desarrollo de nuevo software anunció la existencia del *Borland C++ Builder 4*. Este nuevo paquete puede encontrarse en tres ediciones, la Estándar, la Profesional y la edición Cliente/Servidor.

Para quienes todavía no han oído hablar del *C++ Builder*, podemos decir que se trata de un entorno totalmente visual, que proporciona un desarrollo rápido de aplicaciones (*RAD*) basado en el lenguaje *C++*. Permite programar el interfaz gráfico mediante el arrastre de los componentes que *Borland* ofrece a través de su *VCL* (Librería de Componentes Visuales), para posteriormente asociar el código que ha de ejecutarse como respuesta de los eventos. Veamos sin más dilaciones cuales son las novedades que *Borland* ha presentado en esta versión:

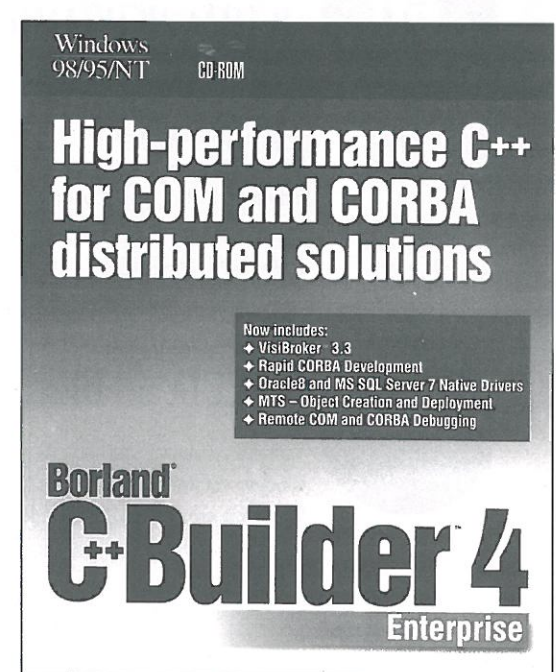
- Permite el desarrollo rápido distribuido con objetos *CORBA* y *COM*. Es la primera herramienta que soporta a la vez los dos estándares.
- Se ha dotado de mayor flexibilidad al compilador para que soporte los proyectos ya desarrollados de *Borland* C++, del *Ansi C++* e incluso de *Visual C++*.
- Se ha mejorado la productividad en el desarrollo permitiendo por ejemplo que el entorno de desarrollo (*IDE*) sea altamente configurable, mover las ventanas flotantes o bloquearlas para lograr un entorno a gusto del programador, o creando nuevos componentes como el *TServiceApplication*.
- Se han incrementado los componentes hasta conseguir mas de 130, entre los nuevos destaca el gran número de servicios creados para Internet (*SMTP, POP, FTP, HTTP, NNTP, HTML y TCP/IP*).
- Se ha añadido al producto un Kit de desarrollo para *MIDAS 2 (Multi-Tier Database Development Service)*.
- Se permiten las últimas mejoras de *Windows 98, 95 y NT.*
- Se han desarrollado nuevas herramientas de depuración que permiten que sea remota para desarrollos distribuidos (*CORBA* y *COM*),.

La edición que nos ha facilitado *Borland* para comenzar a evaluar incluye no sólamente el *C++ Builder 4* sino que además le acompaña la versión 5.5 de *InterBase*, así como el *VisiBroker* un *ORB (Object Request broker*) para *C++* diseñado para facilitar el desarrollo y distribución de la aplicaciones empresariales. Por otra parte, este paquete también incluye el producto *Borland JBuilder 2*, y el ya clásico y mítico *Borland C++ 5.02* con el cual se podrán desarrollar aplicaciones tanto para Windows 3.1x (16 bits) como para *DOS*.

Con todo esto, se ha dotado al entorno *C++ Builder* de las ultimísimas novedades consiguiendo una herramienta capaz de interactuar a cualquier nivel y que evita la engorrosa labor de tener que definir tanto los interfaces gráficos como los objetos con los que trabaja el sistema operativo.

Para todos aquellos que queráis ampliar esta información, podéis visitar el website de *Inprise* **http://www.inprise .com** o más concretamente en esta dirección: **www.borland.com/bcppbuilder** donde se relatan con más detalles todas estas novedades.

En el próximo número de Sólo Programadores aparecerá un exhaustivo análisis de este programa.

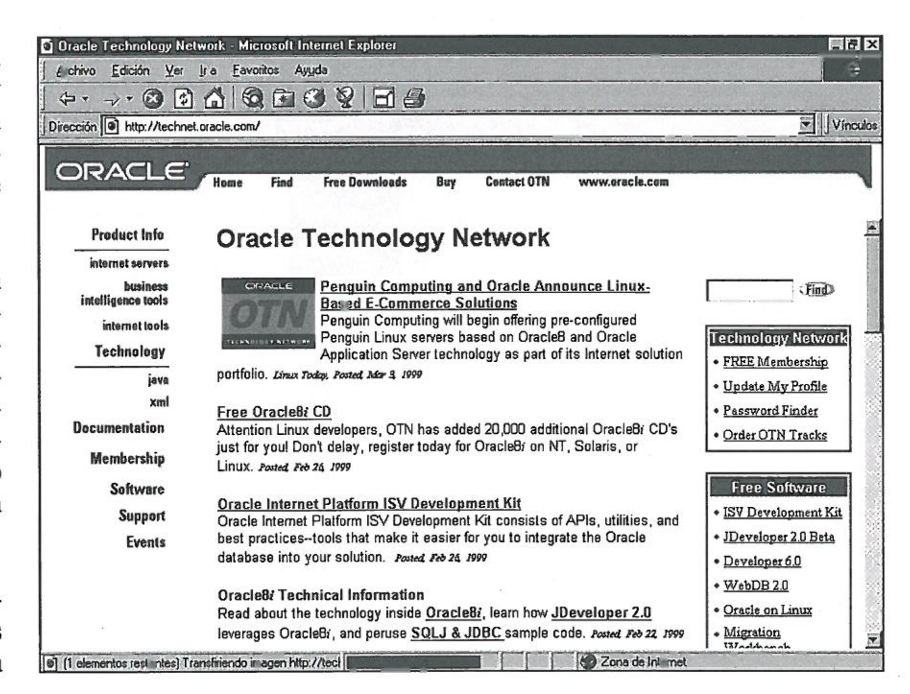


ORACLE CREA NUEVOS ESTÁNDARES PARA PROGRAMACIÓN EN JAVA DE BASE DE DATOS

Oracle ha implantado completamente el estándar SQLJ, en sus herramientas y plataformas de desarrollo. Con esto la compañía consigue ampliar su compromiso con los estándares abiertos de Internet.

La empresa incluye soporte para SQLJ en la actual versión de sus herramientas de programación en Java Jdeveloper 1.1. Con la versión 2.0 de Jdeveloper y la base de datos Oracle8i, los técnicos en desarrollo recibirán como mínimo soporte continuo para SQLJ, tanto en sus entornos de desarrollo como en sus plataformas de servidores.

SQLJ permite incluir directamente instrucciones en código Java. Los desarrolladores podrán centrarse en la funcionalidad empresarial de sus aplicaciones sin necesidad de escribir todos los códigos de acceso de datos de bajo nivel. Se espera que este estándar fomente a que las empresas adopten Java, y que se mejore la cooperación entre fabricantes en relación con este lenguaje. SQLJ fue desarrollado por Oracle junto con IBM y Compaq.



Si lo desea puede ampliar esta información consultando los datos en la siguiente dirección en Internet http://technet.oracle.com

SOFTWARE AG ANUNCIA TAMINO, UN SERVIDOR DE INFORMACIÓN XML

La empresa Software AG acaba de presentar Tamino, un servidor de información esencial para el negocio electrónico.

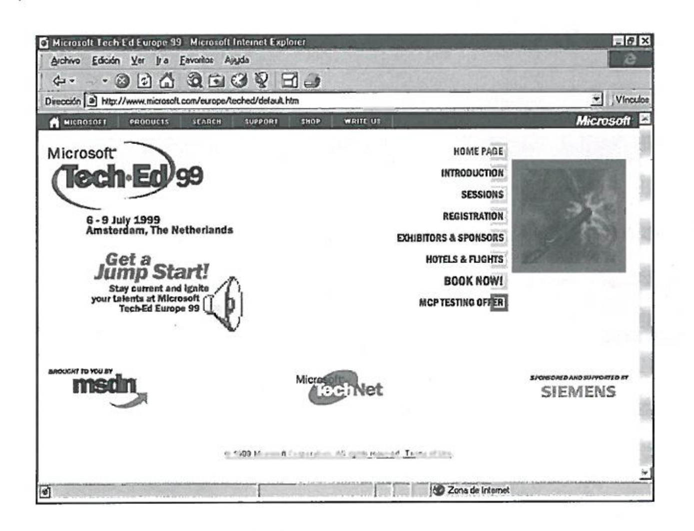
Éste almacena información XML sin tener que convertirla a otra estructura, asimismo, también proporciona este tipo de información con unexcepcional rendimiento para aplicaciones orientadas a transacciones dentro de la propia empresa o en la Web. Igualmente este servidor está preparado para integrar datos de otras bases de datos ya existentes como estructuras del tipo XML.

El nucleo de la tecnología de Tamino se denomina X-Machine y resulta ser la primera en el mundo en almacenar la información XML sin tener que convertirla en otra estructura. Su interfaz es SQL y permite acceder a los datos relacionales los cuales pueden ser integrados sin problema alguno en los documentos del tipo XML.

Tamino es también un completo servidor que actúa como puente entre las aplicaciones existentes y la Web y protege la inversión en los sistemas de información y bases de datos que la empresa posea. Además juega un especial papel en la optimización del almacenamiento de estos datos.

Existe abundante información acerca de este servidor y de todos los productos de la compañia en la dirección www.softwareag.com

TECH.ED 99 REUNIRÁ A MÁS DE 100 COMPAÑÍAS Y A DESARROLLADORES DE MÁS DE 50 PAÍSES



Microsoft nos informa que Tech.ED 99, la mayor conferencia de desarrolladores de software de Europa, tendrá lugar en el Centro Internacional de Congresos y Exhibiciones de Amsterdam del 6 al 9 de julio. Al acto está previsto que asistan cerca de 6.500 representantes de 50 países, convirtiéndose así en el mayor evento organizado por la compañía Microsoft en Europa.

Tech.ED 99 está firmemente consolidada como la principal conferencia para desarrolladores de software y profesionales de las Tecnologías de la Información. El evento, de cuatro días de duración, contará con más de 100 ponentes, que impartirán 270 presentaciones independientes, lo que permitirá a los asistentes aprender sobre las últimas tecnologías de software de Microsoft y de sus socios tecnológicos.

También se ofrecerán durante estos días diferentes exhibiciones de tecnología y herramientas de software con más de 100 compañías participantes. Para obtener más información se puede acceder a la página europea en www.microsoft.com/europe/teched

---

REFUERZO DE LA ALIANZA DE COOPERACIÓN ENTRE CAP GEMINI E IBM

Las compañías IBM y Cap Gemini han suscrito un acuerdo por el cual refuerzan las líneas de cooperación que ambas empresas mantienen desde el año 1997. El objetivo de esta iniciativa es el de conseguir trabajar conjuntamente con el middleware IBM.

Una de las partes más relevantes del acuerdo firmado es que la empresa Cap Gemini creará un Competence Center especializado en las herramientas de desarrollo de las aplicaciones IBM VisualAge.

Este nuevo centro estará ubicado en las instalaciones que Cap Gemini tiene en Utrech (Países Bajos) y tratará de ofrecer a los clientes una creación muy rápida de prototipos, así como también el desarrollo avanzado de aplicaciones utilizando las herramientas VisualAge Generator y VisualAge for Java de IBM.



Puede ampliar información consultando en la siguiente dirección web: www.es.ibm.com

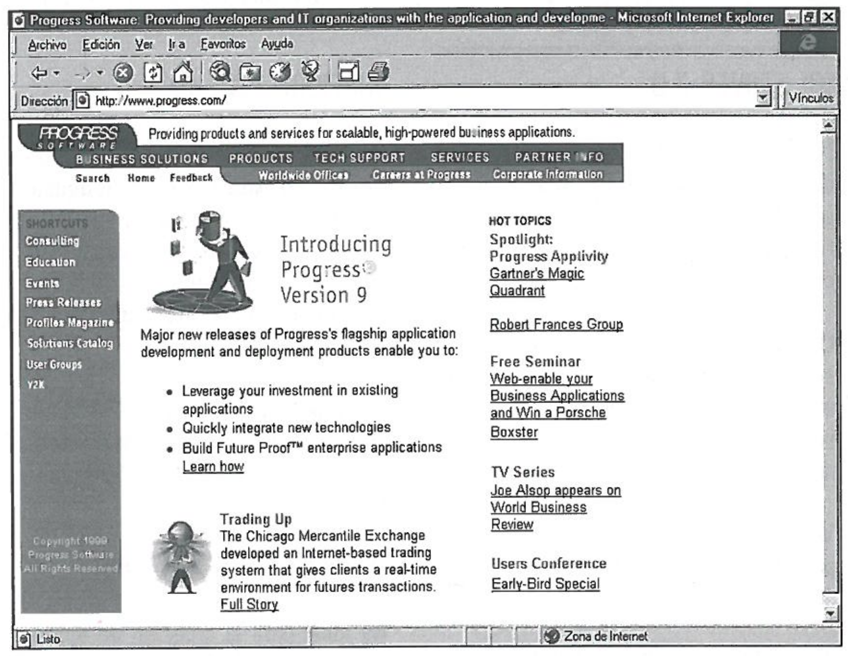
PROGRESS LANZA LA VERSION 9 DE SU CONOCIDA HERRAMIENTA INTEGRADA DE DESARROLLO

La compañía Progress acaba de lanzar al mercado su ultimo producto se trata de Progress Version 9. Hablamos de un amplio conjunto de herramientas integradas de desarrollo, servidores de aplicaciones y productos de bases de datos relacionales.

Esta herramienta proporciona tecnología de elevado rendimiento y altamente escalable, amplia las capacidades de los ISVs de Progress (Fabricantes de Software Independiente) y de los usuarios finales para desarrollar aplicaciones avanzadas, multinivel, con la mayor flexibilidad de implantación.

Además con esta versión se ha conseguido un hito significativo en el marco de la Arquitectura Universal de Aplicaciones de la compañía, que ha sido diseñada para las aplicaciones futuras que trabajan juntas en el seno de una arquitectura basada en estándares presentada en el año 98.

Progress Version 9 está ya disponible en el mercado para Windows 95/98/NT y AS400 así como para aquellas plataformas UNIX más importantes y reconocidas. Si desea ampliar la información sobre este lanzamiento puede hacerlo accediendo a la página Web de la compañía www.progress.com



SE CELEBRÓ EN MADRID LA SEXTA EDICIÓN DE EXPO ORACLE



El pasado mes de febrero se celebró en Madrid la sexta edición de Expo Oracle. Dicha exposición obtuvo su récord de participación reuniendo a un total de 81 expositores.

También cabe destacar que la feria contó con el patrocinio de importantes empresas del sector informático como por ejemplo Hewlett Packard, IBM, Compaq, Siemens y Sun Microsystems.

Durante el evento tuvieron lugar 60 ponencias sobre soluciones para Internet de gestión empresarial, administración pública, desarrollo de aplicaciones, bases de datos, etc..

En este acto también se dieron a conocer los resultados de la Segunda Consulta Oracle, un estudio sobre las tendencias de las Tecnologías de la Información en las empresas españolas.

HERRAMIENTAS DE PROGRAMACIÓN

ENTORNOS DE DESARROLLO

SQL SERVER 7.0

Nueva versión del entorno servidor de bases de datos de MS. Incluye grandes mejoras tanto en la arquitectura interna de la aplicación como en el interfaz.

Ofrece una potencia a la altura de los servidores de bases de datos más avanzados e incluye ayudas muy interesantes para aquellos con menos conocimientos de SQL.

Versión totalmente funcional válida por un periodo de evaluación de 120 días.

DIRECTUPDATE DEVELOPER'S KIT 1.01

Inclusión de actualización automática en nuestros programas. Es un entorno especial para desarrolladores que precisen de aplicaciones auto-actualizables a través de Internet.

El programa chequeará automáticamente la existencia de una nueva versión, la descargará y la instalará. Es una utilidad freeware.

PERL DEVELOPMENT KIT 1.1 BETA 2



Entorno de desarrollo para lenguaje Perl. El kit de desarrollo está compuesto por cuatro componentes. Un debbugger, un compilador, una librería de objetos COM y creador de controles. Con todo esto se ha convertido en uno de los entornos más potentes para el desarrollo en Perl. Se trata de una versión freeware.

SUPERIDE 1.0.1.49

Se trata de un interesante interfaz de desarrollo para cualquier lenguaje de programación. Incorpora un entorno integrado que tiene la capacidad de adaptarse a cualquier tipo de lenguaje, de compilador, de herramienta y diseñador.

Su característica principal es la versatilidad. Incluye corrección sintáctica y plantillas de código predefinidas. Es freeware.

JPADPRO 3.6 BUILD 265

Es ideal para la creación de aplicaciones en Java. JPadPro permite escribir el código, compilarlo, ejecutarlo y testearlo, todo ello por supuesto bajo un mismo interfaz.

Además con él también se pueden localizar los errores de sintaxis y del mismo modo facilitar su resolución. También permite la fácil edición de archivos HTML.

CREATEINSTALL 3.32

Se trata de un Kit de desarrollo de procesos de instalación. Es una utilidad muy interesante que permite distribuir las aplicaciones con un fichero INSTALL.EXE.

Además de no ocupar más de 54 Kb. y que contiene toda la información de la instalación. Dispone de más 14 idiomas y permite todo tipo de operaciones con archivos.

LENGUAJES

VISUAL BASIC

ADVANCED CLOCK CONTROL 1.1.09

Control OCX de reloj analógico fotorealista. Se le pueden añadir diferentes estilos, texturas, sombras, efectos de transparencia y funciones de sonido y alarma. Además soporta más de 100 propiedades distintas de programación que le hacen ser una opción muy flexible.

GRAPHIC VIEWER ACTIVEX COMPONENT 1.0

Control ActiveX que incluye un visor gráfico en la aplicación. Es una de las maneras de añadir un componte tan interesante a los desarrollos.

Soporta múltiples tipos de formatos gráficos, ajuste de tamaño automático, zoom adicional, scrolling variables y gran variedad de opciones. Se incluye el código fuente de un ejemplo de aplicación desarrollada en VB6.

TIMETOWIN 7.01

950 funciones y subrutinas para desarrolladores de Visual Basic. La DLL incluye rutinas de diversas categorías como: arrays, gráficos bitmap, compresión, CRC, fechas, encriptación, list boxes, combo boxes, mail y news, matrices, redes, impresión, objetos,

protección, etc. Una librería imprescindible para desarrolladores avanzados de Visual Basic.

JAVA

123SEARCH 2.0

Creación de motores de búsqueda basados en Java. Incluye un asistente con tres pantallas de introducción de datos para información específica. El programa genera dos JavaScripts que deben incluirse en el fichero HTML que se desea publicar.

JAVABEANS ACTIVATION FRAMEWORK (JAF) 1.0

Herramienta para el desarrollo de JavaBeans. Permite determinar arbitrariamente acceder a una parte de los datos, restringir el acceso a ellos, descubrir las operaciones disponibles en ellos e inmediatamente asigna el componente JavaBean que sea adecuado. Es freeware.

SCRIPT PILOT 1.31

Creación de JavaScripts para la automatización de tareas. Los JavaScripts generados pueden automatizar búsquedas, actualizar y descargar archivos desde sitios FTP y HTT.

También realizar copias de seguridad, etc. Trabaja conjuntamente con el programa NetPad para recoger y procesar datos.

HTML

EZPAD 3.0

Editor de HTML muy poderoso. Ofrece un excelente diseño, interfaz mejorado e incluye un previsualizador de páginas interno.

Viene con una docena de asistentes que permiten realizar casi cualquier tipo de tarea compleja con HTML. Es un programa freeware.

MACROMEDIA FIREWORKS 2.0

Editor gráfico muy potente especializado en proyectos Web. Incorpora diferentes herramientas de texto, de diseño, ilustraciones, edición de imágenes, animación, URL y JavaScript .

Todas ellas permitirán crear todos los elementos habituales de una página Web sin necesidad de utilizar varias aplicaciones a la misma vez para realizarlas.

Soporta los formatos gráficos más habituales, y facilita la optimización de las imágenes en el formato en el que se vayan a exportar.

HOTDOG PRO WEBMASTER SUITE 5.5

Se trata de un profesional, avanzado y flexible gestor/editor HTML. Viene acompañado de los programas Paint Shop Pro, mBed Interactor Lite y Linkbot.

Puede crear plantillas y tags, actualizar fácilmente documentos y sitios enteros, y trabajar en sitios Web actuando en un entorno de trabajo en equipo.

Soporta el procedimiento de arrastrar y soltar para insertar enlaces, imágenes y ficheros de texto y ofrece una previsualización del documento según se está escribiendo.

WEBSCRIPTER 2.021

Hablamos de un importante editor HTML con soporte avanzado para JavaScript. Incluye también menús y botones para crear tags HTML y un excelente soporte para crear scripts JavaScript.

Incluye el código fuente de ejemplo que puede ser modificado para crear eventos de cursor, funciones de escritura de documentos, botones, diálogos, nuevas ventanas, barras de estado e imágenes de enlace.

OTROS

AUTOSQL 2.2

Este interesante lenguaje te permite generar las tablas SQL a partir de otras bases de datos.

Además al mismo tiempo te deja generar las tablas, índices e introducir los datos necesarios, generar el código Delphi 2/4 y C++ Builder 3 de una base de datos dBase, y de tablas Paradox y Access.

Se pueden cortar y pegar los datos y salvarlos como un fichero.

CBUTTONST 2.4

Creación de botones profesionales para C++. CButtonST es una clase derivada de la MFC CButton que permite a sus aplicaciones que usen botones estándar o nuevos diseños realizados con nuevos estilos.

Permite incluir texto e iconos en los botones, iconos de 16 y 256 colores, elegir color y la posibilidad de cambiar el botón cuando el ratón pasa sobre él.

SOCKETTOOLS LIBRARY EDITION 3.0

Se trata de un interesante conjunto de 16 controles VBX y ActiveX para integrar Internet en una aplicación. Los controles que están incluidos son por ejemplo estos:

Domain Name Server, Web Browser Control, un cliente Telnet, POP3, FTP, Remote Execution (RLIB), SocketWrench (WinSock), Ping, Image Viewer, Terminal Emulation, SMPT, NNTP y RAS.

Los desarrolladores que usen Visual Basic 3.0 o Visual Basic 4.0-16 bits pueden usar los controles VBX.

HERRAMIENTAS/ UTILIDADES

2XEXPLORER 0.99

Simulación para W95/98 de Norton Commander de DOS. Permite la sustitución del actual Explorador de Windows 95/98 por un gestor de archivos y directorios mucho más potente que simula al antiguo Norton Commander de DOS.

Para ello muestra un interfaz de doble ventana que permite la interacción con los datos con total simplicidad. freeware.

CACHEMANAGER 2.6

Utilidad para la optimización del caché de Windows. Ayuda a obtener mayor memoria libre a través de la gestión del tamaño mínimo y máximo del caché.

También optimiza los procesos de visualización permitiendo mostrar más rápido el contenido de archivos y carpetas. Es freeware.

CALLCENTER 3.5.8

Excelente paquete integrado de comunicaciones. Programa especialmente indicado para las gestión de las comunicaciones en general y para el control de faxes en particular.

Dispone de opciones para emulación de terminal y soporta ocho tipos distintos de protocolos de comunicaciones.

Hasta ahora era comúnmente llamado FAXCOMponent. Es freeware.

COPERNIC 99 3.0

El software más avanzado para la búsqueda de información en Internet.

Permite la búsqueda simultánea en más de 30 buscadores y robots de Internet. Puede extraer la información de la Web, de las news y de los directorios E-mail usando fuentes como AltaVista, Excite, Yahoo!, HotBot, Infoseek, Lycos, Magellan, Open Text, WebCrawler y DejaNews.

Los resultados son mostrados por orden de importancia y son eliminados los duplicados o los inexistentes. Es freeware.

SHOWDEP 2.50

Esta herramienta nos muestra todas las librerías .DLL que están asociadas a una aplicación. Por cada librería de enlace dinámico, ShowDep muestra la información de la versión, sus módulos de sección, sus módulos de importación y sus módulos de exportación.

Es ideal para examinar las aplicaciones y comprobar si éstas dependen de varios de los ficheros .DLL para su ejecución.

ZOOM SHELL 3.0C

Simulador para Windows de la línea de comandos de Unix. Esta línea de comandos Unix puede ser sustituida por la opción Ejecutar del menú Inicio de Windows 95/98. Zoom Shell soporta alias, scripting, grep y el activado y desactivado del salvapantallas.

Además permite mantener un histórico de las órdenes ejecutadas anteriormente. Es freeware.

REDES

LOCALES

NETPROXY 3.03

Servidor proxy y firewall que permite a una LAN acceder a Internet con una conexión. La conexión a Internet puede ser SLIP, PPP o TCP/IP.

Como la mayoría de las aplicaciones de Internet utilizan el protocolo TCP/IP, es necesario instalarlo en cada equipo de la red y asignar a cada uno de ellos una dirección IP diferente.

El programa permite a los usuarios de la red ejecutar utilidades para Internet como Netscape Navigator, WS_FTP, Telnet, lectores de noticias y clientes de correo electrónico e IRC.

SYGATE 2.0

Permite compartir una conexión a Internet. Incorpora un sistema de limitación de acceso, que impide tanto la visita de intrusos desde Internet, como el acceso a Internet a usuarios de la red local que no están autorizados.

Su instalación es muy sencilla y no es necesario volver a configurar las aplicaciones de Internet que utilizan en cada equipo.

DISTRIBUIDAS

WS FTP SERVER 1.03

Hablamos de un interesante servidor de ficheros para sitios Web. Éste nos ofrece amplias opciones de seguridad configurables y administración de ficheros remotos.

Además de que también nos permite añadir usuarios de forma manual o también importando toda la información desde la base de datos de usuarios de Windows NT o de un fichero de texto.

VISUAL ROUTE 4.0C

Interesante programa para el análisis del estado de las comunicaciones y detección de problemas. Muestra de forma gráfica los resultados, ya que señala las rutas seguidas sobre un mapa del mundo.

Analiza las conexiones IP simultáneamente lo que proporciona su rapidez de funcionamiento.

OTROS

CUENTAPASOS 3.60

Control del gasto telefónico. Programa completo para acceso telefónico a redes que permite al usuario efectuar conexiones a un proveedor de Internet, InfoVía, InfoVía Plus, Retenet o cualquier otro servidor que soporte acceso telefónico a redes; para las llamadas por voz, incluye un marcador telefónico.

Incluye las tarifas vigentes para Infovia Plus, Retenet, Retevisión, UNI2, ONO, MoviStar y Airtel.

DYNAMIP V3.50 BETA 5

Actividades de control sobre los procesos Internet. Ofrece la posibilidad de realizar diversas actividades sobre Internet como actualizar páginas Web, enviar direcciones IP, IPchat, WEBchat, HTTPscan, sincronizar el reloj del ordenador, ejecutar programas automáticamente y otras interesantes funciones.

Incluye soporte para NetMeeting, clientes de correo electrónico, HTTPscan, WEBchat, chats privados, etc..

TRAYPING 1.0

Solución de monitorización de una LAN. El programa realiza periódicamente un "ping" al host remoto y proporciona toda la información necesaria sobre el estado de las comunicaciones. Se trata de una utilidad freeware.

DOCUMENTACIÓN /TUTORIALES

FREEAMP SOURCE CODE 1.1.0

Código fuente en C++ del conocido programa. Código fuente en C++ de uno de los mejores reproductores de audio digital. El paquete incluye los archivos de proyecto originales en Visual C++ 5.0. Es sin duda una buena opción para desarrollar un reproductor mejorado a medida.

INFORMATION RESEARCH KEY 11

Guía HTML para la investigación del fenómeno Internet. Colección de páginas Web que conforman un guía rápida para la investigación a través de una información seria y veraz.

Es una guía de referencia y descripciones ideal para aquellos que necesitan información comercial.

FUENTES DE LOS ARTÍCULOS

En esta interesante sección del CD-ROM os entregamos todos los códigos fuentes que los autores de los artículos nos facilitan, para que os resulte más sencillo, cómodo y agradable.

Con esta información podréis realizar las prácticas, o seguir los ejemplos que han sido planteados en cualquiera de los artículos de las páginas de esta revista.

XML (I)
Ejemplos prácticos realizados en el artículo con XML ./program/XML1/

Creación de un buscador Web (V)
Los fuentes de la práctica propuesta en el artículo ./program/BUSCA5/

Glide (V)
Información adicional al artículo ./program/GLIDE5/

Intranet (II)
Los fuentes de la práctica propuesta en el artículo ./program/INTRANET2/

IMPRESCINDIBLES

ANTIVIRUS

McAfee VirusScan
Panda Antivirus Platinum

GRAFICOS

The Best Icons
LView Pro Image Processor 2.1
Paint Shop Pro 5.01 castellano
ThumbsPlus 3.30s
Xara 3D 3

INTERNET

AutoWinNet 5.5 Beta 2
CuentaPasos 3.6
CuteFTP 2.6
Eudora Light 3.0.6
Go!Zilla 3.3
HomeSite 4.0
mIRC 5.5
URL Organizer 2.0
WebZIP 2.50

MULTIMEDIA

GoldWave 4.02
WinAmp 2.09

NAVEGADORES

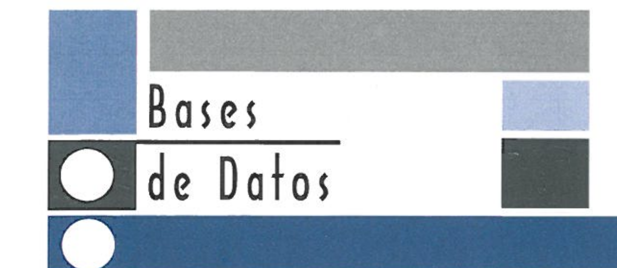
Netscape 4.5
Opera 3.51

RUNTIMES

Runtimes de Visual Basic

UTILS

Adobe Acrobat Reader 3.01
Babylon Translator 2.20
CDR Win
Day Time Organizer 2000
DirectX 6.1
Fix 2000
Windows Commander 3.53
Win32s
WinZip 7.0



Acceso a Oracle 8 desde C++ Builder (y II)

Rafael Corchuelo y Victoria Rus (corchu@lsi.us.es)

En la anterior entrega estudiamos algunas de la nuevas características que incorpora *Oracle 8*, la forma en que debemos preparar nuestro equipo para acceder a él y comenzamos a desarrollar una pequeña base de datos. Ahora aprenderemos a introducir restricciones de integridad.

RESTRICCIONES DE INTEGRIDAD DE REFERENCIA

El siguiente paso consiste en definir las relaciones que existen entre unas y otras tablas utilizando lo que se llaman restricciones de integridad de referencia. Estas nos permiten indicar a *Oracle* que debe velar porque determinadas columnas de una tabla tan sólo tomen valores correctos que aparecen en otra tabla. Por ejemplo, un valor 1 para el campo *COD_NAC* en la tabla **SCOTT. AMIGOS** no tiene sentido a menos que exista una descripción para ese código en la tabla **SCOTT.NACIONAL.**

Estas restricciones se pueden añadir con facilidad colocándonos sobre la pestaña *Referential Constraints* y seleccionando la opción **Object/New.** Al hacerlo el programa pide un nombre para la restricción, podemos dejar el que nos propone por defecto, y en la hoja de propiedades debemos rellenar los dos campos que aparecen. El primero es *Disabled* que indica si la restricción está activa o no. Deshabilitarla aumenta la velocidad a la que se ejecutan las actualizaciones de la base de datos, pero aumenta los riesgos de que la información que contiene sea inconsistente, por lo que no las deshabilitaremos seleccionando la opción **no.** A continuación tenemos que rellenar el campo *Reference Table* para indicar cuál es el nombre de la tabla a la que hacemos referencia. En este caso deberemos seleccionar la tabla **SCOTT.NACIONAL** y a continuación en la pestaña *Columns* añadir el nombre de la columna por la que queremos hacer referencia a esta tabla, *COD_NAC* en este caso.

Conviene indicar que tan sólo es necesario añadir la restricción en la tabla de amigos, no en la de nacionalidades. La razón es que todos los códigos de nacionalidades que aparecen en la tabla de amigos deben estar definidos en la tabla de nacionalidades, pero no todos lo que aparecen en ésta deben ser utilizados en la tabla de amigos.

OTRAS RESTRICCIONES DE INTEGRIDAD

Además de las restricciones de integridad de referencia, *Oracle* permite añadir otras restricciones libres a nuestras tablas. A continuación se recogen las más interesantes:

- **CAMPO IS NOT NULL.** Indica que la columna *CAMPO* tiene que tener siempre un valor correcto. En nuestro caso es fundamental que nunca sean nulos los campos *NOMBRE* en la tabla **SCOTT .AMIGOS** y los dos campos de la tabla **SCOTT .NACIONAL.** No obstante, *Oracle* añade esta restricción de forma automática a todos aquellos campos que sean clave de una tabla, por lo que realmente tan sólo tendremos que añadir la restricción al campo *DESCRIP* en la tabla **SCOTT.NACIONAL.**
- **CAMPO IN (*lista de valores*).** Indica que el valor del campo debe ser uno de los que aparecen en la lista de valores. Por ejemplo, sobre un campo que representa el sexo de una persona podemos imponer la restricción **SEXO IN ('H', 'M').**

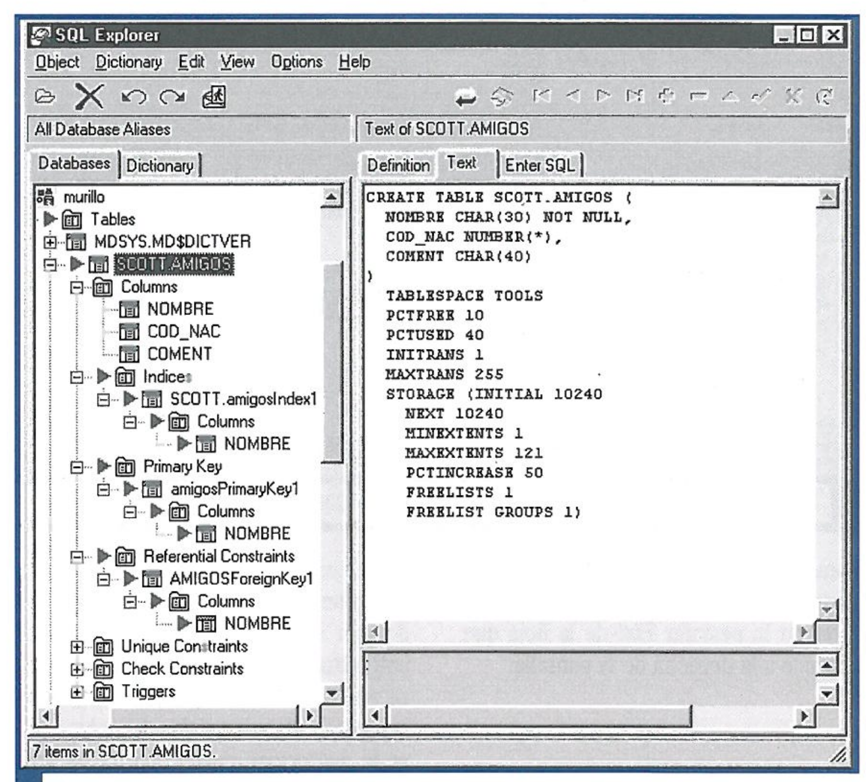


Figura 1. Aspecto de la ventana de SQL Explorer después de haber creado nuestra base de datos de ejemplo.

La paleta de componentes proporciona diversos elementos de acceso a las bases de datos

- **CAMPO BETWEEN valor$_1$ AND valor$_2$.** Obliga a que el valor que contiene el campo esté entre los límites marcados por los dos valores indicados. Por ejemplo, si deseamos que los códigos de nacionalidad sean enteros en el rango (1-50), podemos usar como restricción **COD_NAC BETWEEN 1 AND 50.**
- **EXP**: especifica una condición que cada registro de la tabla debe satisfacer. Por ejemplo, si deseamos que todas las descripciones de nacionalidades estén escritas en mayúsculas, podemos usar la restricción **DESCRIP=UPPER (DESCRIP).**

Para crear este tipo de restricciones debemos colocarnos en la pestaña *Check Constraints* y usar la opción **Object/New.** Al hacerlo en la hoja de la derecha aparecen dos campos que debemos rellenar de forma obligatoria. En el primero debemos escribir la restricción, una de las que hemos comentado anteriormente, y en el segundo debemos seleccionar si queremos que *Oracle* la active o no. Como hemos comentado anteriormente, lo más recomendable es activarlas puesto que es la mejor forma de garantizar la consistencia de los datos. La sobrecarga que se produce al activarlas no suele ser demasiado grande en comparación con los beneficios que proporcionan.

ACTUALIZACIÓN DE LOS CAMBIOS EN EL SERVIDOR

Una vez creadas todas las tablas, las columnas, los índices, las claves y las restricciones, ya sólo falta actualizar los cambios que hemos hecho en la estructura de la base de datos en el servidor. Para ello tenemos que colocarnos sobre la pestaña principal que tiene el nombre de la base de datos, la primera en el árbol, y seleccionar la opción **Object/Apply.**Se nos informará de que los cambios que se van a realizar pueden llevar consigo pérdida de información en caso de que las tablas ya tengan algo, pero como de momento no hemos introducido ningún dato, podemos decirle que sí sin preocuparnos de nada más. Como norma general, le recomendamos que piense con detenimiento la forma de diseñar su base de datos, ya que modificar su estructura cuando ya tiene datos puede dar lugar a pérdidas irremediables de información.

La figura 1 muestra una pantalla del *SQL Explorer* en la que hemos realizado todas las operaciones que hemos ido indicando sobre la base de datos. Si desea en algún momento saber cuál es la instrucción *Oracle* que debería haber tecleado en la herramienta *SQL*Plus*

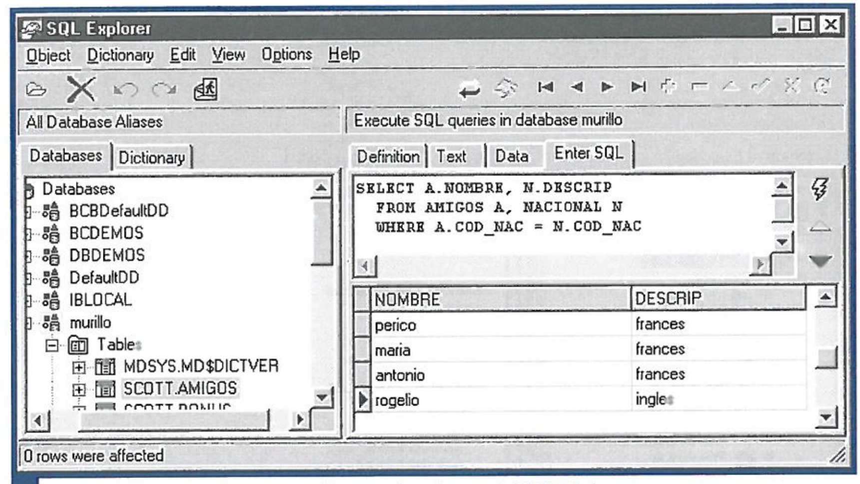


Figura 2. Ejecución de una consulta compleja dentro del SQL Explorer.

para crear una tabla, una columna o una restricción de integridad, puede consultarla en la pestaña *Text* de la hoja que aparece a la derecha de la pantalla.

CÓMO INTRODUCIR O CONSULTAR DATOS

Introducir datos en nuestras tablas recién creadas no puede ser más simple. Tan sólo debemos colocarnos en el árbol sobre la pestaña que tiene el nombre de la tabla que deseamos rellenar, **SCOTT.NACIONAL** por ejemplo, y en la hoja de la parte derecha seleccionar la pestaña *Data*. Ésta es una rejilla de datos en la que podremos introducir tantos registros como deseemos.

La forma más adecuada de introducir datos es en modo transacción. Para seleccionar este modo tan sólo tenemos que pulsar el botón **Begin Transaction** que aparece en la barra de botones. Al hacerlo, todos los registros que insertemos o modifiquemos en las tablas no se insertan de forma automática, sino en bloque cuando volvamos a pulsar este botón. Si por el contrario, vemos que hemos cometido algún error y queremos descartar los cambios que hemos realizado, tan sólo tendremos que pulsar el botón **Rollback Transaction** que aparece a su lado. Como ejemplo compruebe mientras rellena una tabla de amigos qué es lo que ocurre cuando en un registro teclea el código de una nacionalidad inexistente. Dado que antes definimos una restricción de integridad referencial, *Oracle* previene la introducción de registros en la tabla de amigos cuya nacionalidad no esté definida en la tabla de nacionalidades y nos informa de ello mediante una ventana con un mensaje de error.

Las restricciones permiten mantener la integridad de los datos

Para realizar consultas podemos utilizar *SQL* en la hoja *Enter SQL* que se encuentra justo al lado de la hoja *Data*. Por ejemplo, si queremos consultar todos los amigos registrados que tienen nacionalidad **2** introducimos:

```
SELECT *
    FROM AMIGOS
    WHERE COD_NAC = 2
```

La figura 2 muestra un ejemplo más complejo en el que hemos recuperado de forma conjunta los nombres de nuestros amigos y sus nacionalidades. Aunque en nuestro ejemplo siempre hemos usado mayúsculas, no se establece diferencia entre unas y otras más que a la hora de definir el nombre de las tablas o los campos. Por otra parte, tampoco es preciso hacer referencia a las tablas usando el nombre **SCOTT.AMIGOS** ó **SCOTT.NACIONAL**. El nombre de usuario se lo coloca de forma automática *SQL Explorer*.

COMPONENTES DE ACCESO A ORACLE

En las secciones anteriores hemos estudiado de qué forma se configura nuestro *PC* para poder acceder a *Oracle* y de qué manera podemos crear una pequeña base de datos que cuenta con casi todos los elementos que ofrece *Oracle 8*. Para ilustrar de qué forma se puede acceder a las tablas que hemos creado desde dentro de nuestras aplicaciones en *C++ Builder* vamos a crear un pequeño programa. Para empezar debemos crear un nuevo proyecto con la opción **File/New Application.** Con esto obtendremos la ficha principal (*Form1*), donde visualizaremos los datos, y su fichero de código *C++* asociado **unit1.cpp** (con la tecla **F12** podemos cambiar entre la ficha y el código).

TABLAS Y FUENTES DE DATOS

Los objetos de acceso a las tablas (*TTable*), consultas (*TQuery*), etc., los colocaremos en un módulo de datos. Para crear uno debemos elegir la opción **File/New Data Module**, con lo se crea una ficha con nombre *DataModule2* que tendrá asociado el fichero de código **unit2.cpp**. Dentro de este módulo debemos colocar dos componentes de la clase *TTable* de la página *Data Access* de la

paleta de componentes. Lo primero es asignar un valor a la propiedad *DatabaseName* de las dos tablas mediante el *Object Inspector* que aparece con la tecla **F11**. *DatabaseName* ofrecerá en una lista desplegable todos los alias definidos con *SQL Explorer*. Para especificar la tabla a la que queremos acceder utilizaremos la propiedad *TableName*. A continuación, simplemente colocando la propiedad *Active* con el valor **true** ya podremos insertar, modificar y borrar datos de la tabla desde dentro de *C++ Builder*.

No olvides configurar las consultas SQL de forma que se ejecuten en el servidor Oracle

Para visualizar los datos colocaremos dos componentes de la clase *TDBGrid*. El problema es que no podemos conectar estos componentes directamente a las tablas, sino que debemos insertar dos componentes intermedios del tipo *DataSource* en el módulo de datos *DataModule2*. La única propiedad importante de estos componentes es *DataSet* con la que indicamos cuál es la tabla de datos a la que permiten acceder.

Para visualizar los datos en cada rejilla sólo hay que elegir en la propiedad *DataSource* de cada una el componente DataSource asociado a cada tabla que se pretende visualizar.. Esto es debido a que la ficha principal y el módulo de datos están en distintos ficheros, así que habrá que incluir en la cabecera del código asociado a la ficha principal (fichero **unit1.cpp**) el fichero de cabecera del módulo de datos (**unit2.h**). Esto lo podemos hacer de dos formas: (1) colocándonos en la ficha principal, pulsando **F12** para ver su código y escribiendo la línea **#include "unit2.h"** en el lugar que se indica en la figura 3 utilizando la opción **File/Include Unit Header** del menú principal. Para ver los datos tan sólo es preciso colocar la propiedad *Active* al valor **true**.

NAVEGANDO POR LOS DATOS

Para comprobar que todo va bien podemos ejecutar nuestra aplicación inmediatamente con la opción **Run/Run** del menú principal. Ahora bien, cuando el número de datos almacenados es grande resulta muy recomendable añadir un componente de la clase *TDBNavigator* para movernos por ellos. La propiedad más importante de las barras de navegación es de nuevo *DataSource*, que unirá cada barra con el *DataSource* de una de las dos tablas.

CONSULTAS EN SQL

Para poder ejecutar instrucciones *SQL* se utilizan componentes de la clase *TQuery*, que se encuentran en la solapa *Data Access* de la paleta de componentes de *C++*. Estos permiten ejecutar cualquier consulta en el lengua *SQL* que soporta *Oracle 8*, pero para sacarle toda su potencia al servidor no debemos olvidar colocar el parámetro *SQLQRY MODE* al valor **SERVER**. De lo contrario las consultas se ejecutarán de forma local en su *PC* y no aprovechará el servidor más que para almacenar datos.

A modo de ejemplo, vamos a colocar un *TQuery* en el módulo de datos que nos permita conocer cuál es la nacionalidad de cada uno de nuestro amigos. Para ello basta con colocar la propiedad *DatabaseName* a **murillo** e introducir en la propiedad *SQL* la siguiente instrucción:

```
SELECT A.NOMBRE, N.DESCRIP
        FROM SCOTT.AMIGOS A,
       SCOTT.NACIONAL N
        WHERE A.COD_NAC = N.COD_NAC
```

Para visualizar los datos que produce esta consulta debemos asociarle un *DataSource*

```
void __fastcall TForm1::Button1Click(TObject
        *Sender)
        {
         DataModule2->Query1->Close();
         DataModule2->Query1->Prepare();
         DataModule2->Query1->Open();
        }
```

que ejecute las instrucciones mencionadas al ser pulsado *OnClick*.

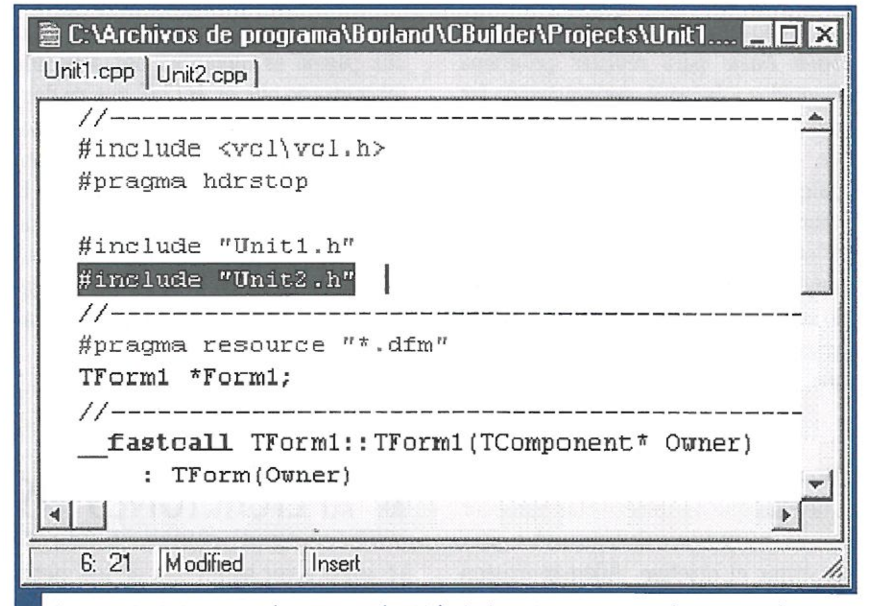


Figura 3. La ficha principal no conoce al módulo de datos hasta que no incluyamos en el código de dicha ficha el archivo de cabecera del módulo.

Aplicaciones IIS (I)

Jordi Agost (agost@eup.udl.es)

En este artículo estudiaremos las aplicaciones *IIS*, acrónimo del producto de *Microsoft, Internet Information Server*. Dichas aplicaciones son las que residirán en un *web* y se utilizarán para procesar las peticiones de un navegador. Realizaremos el proceso en el servidor y ejecutaremos en él el código *Visual Basic* asociado.

VENTAJAS DE VISUAL BASIC E INTERNET

En este artículo seguimos aprovechando nuestros conocimientos de *Visual Basic* para realizar programa orientados a *Internet*, aprovechando las ventajas que ofrece esta herramienta y que exponemos a continuación. En cualquier caso, para realizar estos desarrollos para *Internet* disponemos de otras muchas opciones fuera del mundo de *Visual Basic*, como por ejemplo programas *CGI*, páginas activas de servidor (*Active Server Pages*), *ActiveX*, *Java*, etc.

El hecho de utilizar *Visual Basic* ofrece algunas ventajas, siendo la primera de ellas que no se necesitan aprender nuevas secuencias de comandos para conseguir el objetivo. Además existen una serie de temas mucho más asequibles con *Visual Basic*, como por ejemplo la depuración, hecho que puede llegar a ser algo difícil con otro tipo de soluciones como con las aplicaciones *CGI*. También desde *Visual Basic* podremos separar la interfaz del usuario del código de una página.

Además en *IIS* podemos llegar a reutilizar páginas en diferentes contextos, ya que la información para explorar una página se almacena por separado (al contrario que en *HTML* que va dentro del propio código), siendo así posible reutilizar la misma página en muchos sitios tan sólo cambiando la información del contexto. Y como punto final podemos decir que el tiempo de descarga es menor ya que en ningún caso necesitaremos descargar componentes de gran tamaño.

APLICACIONES IIS

Vamos a ver ahora uno de los métodos mediante los cuales podemos aprovechar nuestros conocimientos de *Visual Basic* en la programación de *Internet*: las aplicaciones *IIS*. Podemos decir que una aplicación de este tipo (aplicación de *Internet Information Server*), es una aplicación que combina código *HTML* y código *Visual Basic* compilado, en una aplicación basada en un explorador.

Al igual que una *ASP* también reside en un servidor *web*, donde recibe peticiones del explorador de un usuario y devuelve páginas *HTML* de acuerdo con esas peticiones. Es decir, empleamos código *HTML* para presentar una interfaz de usuario y código *Visual Basic* compilado para responder y procesar las demandas que hace el usuario a través de su navegador. Resumiendo, el usuario tan sólo verá una sucesión de páginas *HTML*, pero el programador sabe, que una aplicación *IIS* consta de un objeto especial llamado clase *web* o clase de *web*, y dicho objeto a su vez contiene otros subobjetos denominados elementos de *web*. Dicho objeto, *clase web*, es como el cerebro de la aplicación *IIS*, la parte encargada de recibir los datos enviados por el usuario a través del navegador y se encarga posterior-

mente de enviar los datos que formarán la respuesta de la aplicación ante la demanda del navegador. Los datos que se devolverán serán aquellos elementos de *web* (objetos derivados de la clase *web*), que serán páginas *HTML*, así como otros elementos.

REQUISITOS DEL SISTEMA

Para desarrollar aplicaciones *IIS* necesitaremos tener instalado en nuestro equipo la versión 4.0 o posterior del explorador *Internet Explorer* y además un servidor *web* que sea capaz de trabajar con páginas *ASP*. El programa en sí dependerá del sistema operativo que tengamos instalado. Si tenemos *Windows NT Server 4.0* o *Windows NT WorkStation 4.0* (los dos con el *Service Pack 3*), entonces deberemos tener *Internet Information Server 3.0* o posterior con las páginas *ASP* también instaladas.

Una aplicación IIS combina código HTML y código Visual Basic compilado

En cambio, si estamos usando el sistema operativo *Windows 95/98*, deberemos usar el programa Personal *Web Server 3.0* o posterior también con la opción de las páginas *Active Server*. Para el usuario o equipo final tan solo será necesario un ordenador que tenga instalado un navegador cualquiera.

IIS FRENTE A ASP

En el artículo anterior vimos el uso de *VBScript* y las aplicaciones de páginas activas de servidor. A primera vista puede parecer que son aplicaciones paralelas, con un mismo fin, pero dicho parecido es tan sólo superficial. Aunque

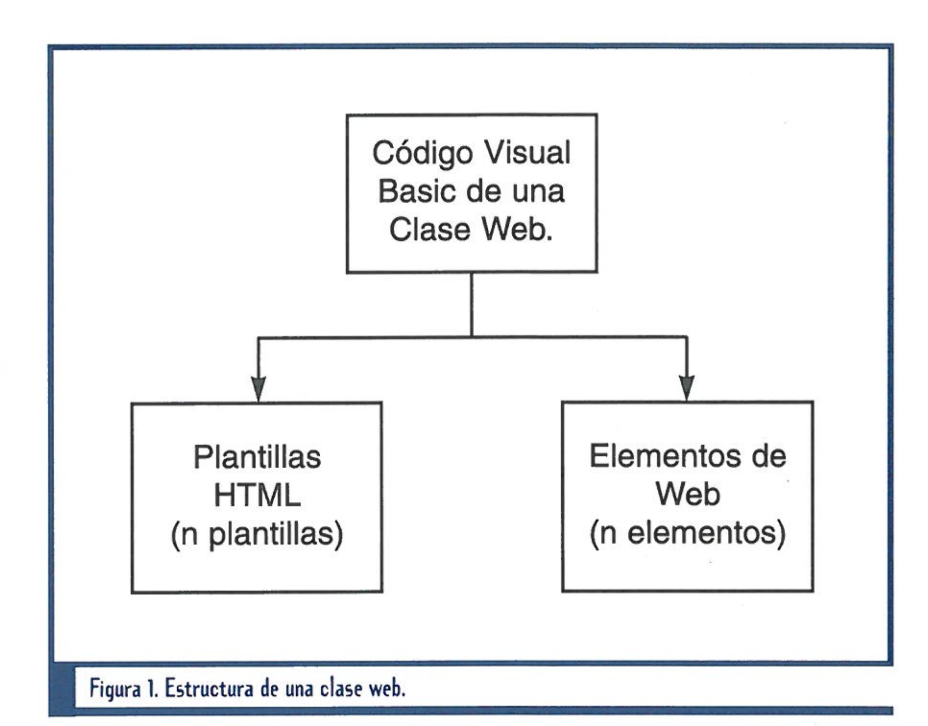


Figura 1. Estructura de una clase web.

los dos tipos de aplicaciones pueden mostrar sitios *web* de contenido dinámico y además realizan su procesamiento en el servidor *web* en vez de en el ordenador del cliente, las páginas *ASP* están más dirigidas hacia los programadores de *scripts* permitiendo, tal y como vimos en el artículo anterior, mezclar código *HTML* y código *VBScript*. Por el contrario las aplicaciones *IIS* se encuentran más orientadas a programadores de *Visual Basic*, programadores que deseen crear aplicaciones basadas en *web*, en lugar de páginas *web*.

APLICACIONES IIS FRENTE A APLICACIONES "EXE ESTÁNDAR"

Si las diferencias existentes entre las aplicaciones *IIS* y las *ASP* eran de tipo superficial, no ocurre lo mismo cuando la comparación se efectúa con aplicaciones estándar. Si lo que queremos es trabajar con aplicaciones *IIS* tendremos que romper el esquema mental sobre la construcción y el proceso de las aplicaciones típicas de *Visual Basic*, ya que ambos tipos de aplicaciones están basadas en esquemas diferentes. Para empezar diremos que las interfaces de una aplicación *IIS* son normalmente diversas páginas *HTML*. Además *Visual Basic* no generará ningún código *HTML*, es decir, no realizará la interfaz de nuestro programa, por eso, será necesaria la presencia de una segunda persona (aparte del programador o en su defecto el programador mismo) que realice las páginas *HTML*. Otra diferencia será que en el momento de ejecutar el código de una aplicación de *Visual Basic* estándar podremos observar el programa sin más, dentro del entorno mismo de *Visual Basic* (si estamos compilando) o dentro de *Windows* (programa ya compilado). En cambio para observar una aplicación *IIS* necesitaremos un explorador *web*.

CLASES WEB

El componente principal dentro de una aplicación de tipo *IIS* es una clase *web*. Se trata de un componente

más de *Visual Basic*, con la diferencia de que reside en un servidor *Web* y su misión consiste en responder a la información que necesita un navegador. Se compone de elementos de *web*, que son aquellos que se pueden devolver como respuesta a una petición de un navegador o petición *HTTP*. Dichos elementos puede ser páginas *HTML*, archivos *MIME* (*Multimedia Internet Mail Extension*), etc., junto con su código asociado. Una de las características de una clase *web* consiste en que por cada navegador que se conecta al servidor, es decir, por cada conexión establecida, existirá una y tan sólo una clase *web* asociada a dicho navegador (*Visual Basic* creará tantas instancias lógicas de una clase *web* como se precisen). Si la conexión se rompe, entonces la instancia se destruye, pero mientras no suceda esto, la clase *web* puede mantener el estado o la información entre varias peticiones de un mismo cliente, hecho que será de gran utilidad, como veremos posteriormente.

CONTENIDO DE LA CLASE WEB

Antes de proseguir con el contenido de la clase *web* debemos comentar que normalmente en un proyecto de tipo *IIS* sólo se usa una (recordemos que es creada automáticamente al iniciar la programación de la aplicación). De todas formas, en ciertos proyectos pueden llegar a utilizarse más clases de este tipo, especialmente si éstas provienen de una reutilización de otras clases *web*, utilizadas en otras aplicaciones.

Una clase web es un elemento de Visual Basic pero que reside en un servidor web

Todos los componentes de una clase *web* son conocidos con el nombre de "elementos de *web*". Estos normalmente proporcionan elementos que puede mostrar el navegador del usuario y los correspondiente eventos para dichos elementos.

Existen dos tipos de elementos de clase *web*: los archivos *HTML* de plantilla y los elementos personalizados de *web*. Vamos a ver a continuación una pequeña descripción de los dos elementos constituyentes de la clase *web*:

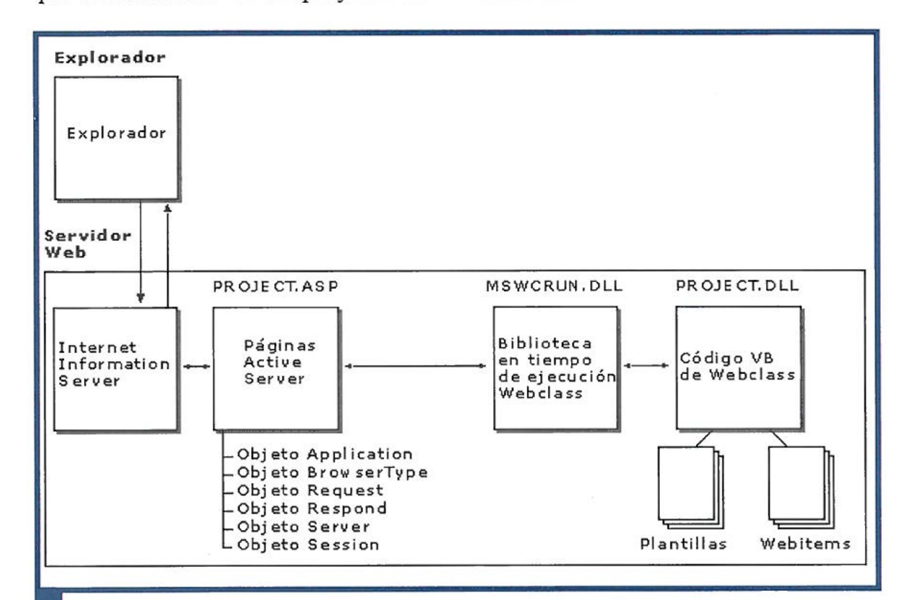


Figura 2. Gráfico de la ayuda de Visual Basic donde se muestra la estructura de una aplicación Internet Information Server.

ARCHIVOS HTML DE PLANTILLA

Se trata de páginas *HTML* asociadas a una clase *web*, de tal forma que cuando la clase *web* recibe una petición por parte del usuario, o mejor dicho del navegador, se envían estas páginas al navegador para que las visualizce. La única diferencia entre dichas páginas y las páginas *HTML* normales sería que éstas poseen unas zonas especiales que en un momento dado pueden sustituirse para dar así una respuesta personalizada al navegador del usuario.

ELEMENTOS PERSONALIZADOS DE WEB

Dichos elementos son elementos controladores de eventos a los que el navegador llamará cuando se cargue la página o se seleccione un determinado elemento *HTML*. En ese momento el controlador de eventos (los elementos personalizados de *web*) pueden decidir si establecen una determinada respuesta o bien si pasan el procesamiento a otros elementos de la clase *web*. Dichos elementos no posen *HTML*.

Debemos especificar que cada uno de los dos elementos de las clases *web* que acabamos de ver posee sus propios eventos. Dichos eventos serán procesados por la clase *web* cuando en el explorador ocurran ciertas acciones por parte del usuario que desencadenen dichos eventos. Usaremos *Visual Basic* Estándar para escribir la respuesta a dichos eventos, uniendode este modo nuestro código al evento que se haya producido en el navegador.

PARTES DE UNA APLICACIÓN IIS

Una aplicación *IIS* consta principalmente de una o más clases *web*. También encontraremos plantillas *HTML* (que como ya hemos visto, son páginas *HTML* diseñadas para que algunas de sus partes puedan ser reemplazadas con contenido dinámico, y actuar así a modo de plantillas) con sus correspondientes eventos. También podemos encontrar elementos personalizados de *web* junto con sus correspondientes eventos. Otro elemento constituyente de una clase *web* será un archivo *ASP* (*Active Server Pages* ó páginas activas de servidor). Además deberemos tener la siguiente librería **MSWCRUN.DLL**, que es un componente de la clase *web* y su tarea consiste en procesar las peticiones del navegador. Finalmente una aplicación de este tipo requerirá una librería *DLL* que permita empaquetar todo el código de *Visual Basic* dentro de ella.

Cabe añadir que bastantes elementos de los anteriormente mencionados se generan automáticamente al crear el proyecto de clase *web*. El archivo *ASP* mencionado en el párrafo anterior también se generará automáticamente, y tendrá el nombre que le hayamos asignado a la propiedad *NameInURL*. En cuanto a la librería *DLL* también se genera de forma automática a través del proceso de compilación.

ESTRUCTURA FÍSICA DE LAS APLICACIONES IIS

Al igual que otras aplicaciones, de *Visual Basic*, una aplicación *IIS* posee módulos de código y un diseñador visual. Los objetos de la aplicación *IIS* se almacenan en archivos de texto sencillo que contienen el código fuente de la clase de *Web*, la configuración de eventos y propiedades, y los elementos de la clase *web*. *Visual Basic* utiliza la extensión **.dsr** para estos archivos. Aparte del archivo **.dsr**, *Visual Basic* genera un archivo **.dsx** que contiene información binaria sobre la aplicación (tal y como sucede con los formularios (*frm* y *frx*).

LA CREACIÓN DE UNA APLICACIÓN IIS

Para crear una aplicación de tipo *IIS* desde *Visual Basic* actuaremos de forma análoga a como lo hacemos normalmente con proyectos estándar. En primer lugar iniciaremos un nuevo proyecto y seleccionaremos **Aplicación IIS**. En segundo lugar guardaremos el proyecto, ya que en caso contrario no podremos continuar (no nos dejaría añadir ciertos elementos posteriormente). Seguidamente añadiremos a la clase *web*, todos los elementos *web* que sean necesarios, bien sean plantillas *HTML* o personalizados de *web*. Agregaremos más tarde los eventos personalizados y escribiremos código para los eventos.

Las clases Web poseen eventos de tres tipos: estándar, de plantilla y personalizados

A continuación añadiremos, si son necesarios, otros módulos de código u objetos de tipo *WebClass*. Seguidamente depuraremos la aplicación y la visualizaremos desde varios exploradores. Por último lo que haremos será compilar la aplicación y la distribuiremos en la red.

CLASES WEB Y ARCHIVOS ASP

Vamos a analizar la relación entre las clases *web* y los archivos *ASP*. Como ya hemos mencionado, cada una de las clases *web* de una aplicación *IIS* (recordemos que normalmente cada aplicación *IIS* tiene una clase *web* pero puede tener más de una), lleva asociado un archivo de tipo página activa de servidor o archivo *ASP*. Dicho archivo es generado automáticamente durante el proceso de compilación de la aplicación *IIS*. Su misión consiste en alojar la clase *web* en el servidor *Web*, y generar el componente en tiempo de ejecución cuando se ejecute por primera vez la aplicación.

En el gráfico apreciamos las relaciones que estamos explicando. Cada clase *web* tendrá su propio archivo *ASP* y a su vez una clase *web* tendrá varios elementos de *web* asociados. La ruta que tenga el archivo *ASP* actuará como la dirección *URL* base de la clase *web* y sus correspondientes elementos.

Imaginemos por ejemplo, que nuestro proyecto *IIS* se denomina **ProyectoPrueba** y tenemos una clase *web* que tiene el nombre de **ConsultaPrueba**. Entonces en la propiedad *NameInURL* de la clase *web* pondríamos el siguiente valor: **Proyecto Prueba_ConsultaPrueba** con lo que le estamos informando a *Visual Basic* que cuando cree la página *ASP* (recordemos que esto se hace durante el proceso de compilación y/o depuración) ésta se llame **ProyectoPrueba_ ConsultaPrueba.asp** y la almacenamos en un directorio concreto llamado **Temp**. Entonces la dirección *URL* base de la clase *web* sería:

http://www.loquesea.com/Temp/
ProyectoPrueba_ConsultaPrueba.asp

Esta sería la dirección web a la cual se conectaría el usuario al iniciar la aplicación.

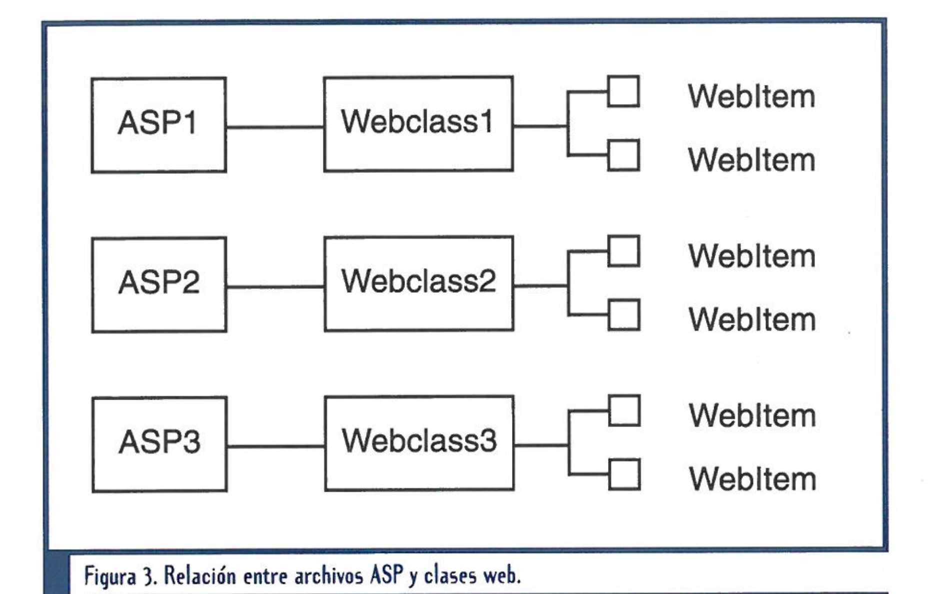


Figura 3. Relación entre archivos ASP y clases web.

EL MODELO DE OBJETOS EN LAS APLICACIONES IIS

Tal y como hemos visto una aplicación de tipo *IIS* se aloja dentro de un archivo *ASP*, esto le permite utilizar varios de los objetos de las *ASP*. Es decir, la clase *web* puede utilizar objetos de las páginas activas de servidor para manipular o extraer alguna información de una página *HTML*. Algunos de los objetos los vimos en el anterior artículo (como por ejemplo el objeto **Session, Response, Request y User**) y otros los vamos a ver a continuación en un pequeño resumen.

- El objeto **Request** recibe peticiones de parte del explorador.
- El objeto **Response** enviará información al navegador.
- El objeto **Session** mantendrá información sobre la sesión actual del usuario y sobre el estado actual.
- El objeto **Application** administrará estados compartidos por varias instancias de la clase *web*.
- El objeto **Server** creará otros objetos y definirá las propiedades específicas del servidor que puedan afectar al procesamiento.
- El objeto **BrowserType** determinará la capacidad del explorador del usuario, para así tomar mejores decisiones de procesamiento.

LOS EVENTOS EN APLICACIONES IIS

Normalmente el mayor porcentaje de instrucciones de una aplicación *IIS* se corresponde con los procedimientos de eventos de los elementos *web* de la aplicación, que son los utilizados por *Visual Basic*, y son los que le indican qué debe hacer cuando el explorador lanza una petición hacia un elemento específico.

Para conservar el estado entre dos peticiones sucesivas podemos utilizar el objeto WebClass

Algunas de las tareas que pueden realizar estos procedimientos son las de obtener datos del usuario, realizar sustituciones de datos en plantillas *HTML*, abrir una conexión de base de datos, crear conjuntos de registros y manipular datos, desplazar al usuario a otro elemento de web, mostrar otra página de la aplicación, escribir el contenido de una plantilla *HTML* en el explorador, generar código *HTML*, etc..

EVENTOS EN UNA CLASE WEB

Veamos con un poco más de detenimiento los eventos. Como muchos de los elementos de *Visual Basic*, las clases *web* también poseen eventos, que podemos dividir en tres clases: eventos de tipo estándar, eventos de plantilla y eventos personalizados.

EVENTOS ESTÁNDAR

Son eventos predeterminados, es decir los que ya vienen por defecto en la clase *web*. Se aplican sobre las plantillas y los elementos personalizados de *Web*, y sólo los podemos ver en la lista Procedimiento de la ventana del editor de código.

EVENTOS DE PLANTILLA

Son eventos generados por *Visual Basic*, una vez éste ha analizado el archivo de plantilla *HTML*. Dichos eventos actuarán sobre las plantillas y los podremos ver en la ventana del diseñador y en la ventana del editor de código después de conectar.

EVENTOS PERSONALIZADOS

Son los que el programador añade según sus necesidades. Aparecen en la ventana del editor de código, para que podamos introducir el código que deseemos y también en la ventana del diseñador. Sus efectos se verán sobre los elementos personalizados de *web* o sobre las plantillas.

Lo primero que estudia la aplica ción *IIS* es si la petición especifica un evento o nó. Seguidamente mirará si el evento corresponde a la clase *web*. Suponiendo que así sea, la aplicación activará el evento (los irá mirando uno a uno hasta encontrar el procedimiento de evento y el elemento de *web* apropiados). Si los eventos no pertenecen a una clase *web*, entonces se activará el evento *UserEvent* para la plantilla.

Es mejor utilizar direcciones relativas para evitar modificar y compilar todo el código

Si la petición proveniente del explorador no se corresponde con un evento, y tampoco especifica ningún elemento de *web*, entonces el sistema iniciará el archivo *ASP* correspondiente y activará el evento *Start* de la clase *web*. Suponiendo, en caso contrario, que la petición especifique un elemento de *web*, la clase *web* activará el evento *Respond* para la plantilla.

EL DISEÑO EN WEB

Ahora que ya conocemos los fundamentos de una aplicación de tipo *IIS* vamos a detenernos un momento en la descripción de la aplicación misma para ver unas recomendaciones sobre el diseño de aplicaciones sobre *Internet*. Desde el punto de vista de la programación de una aplicación basada en *web*, ésta ofrece mayores dificultades que la programación de una típica aplicación basada en formularios. En una aplicación típica, el programador va guiando más o menos al usuario a través de la navegación por los diferentes formularios, incluso ésta puede ser bastante rígida como en el caso de los asistentes, donde normalmente las únicas opciones que se permiten son las de avanzar hacia delante o bien retroceder. Y normalmente antes de finalizar la aplicación se controla si hay cambios, o si estos deben guardarse. En cambio en una aplicación basada en *web*, el usuario puede en cualquier momento cerrar la aplicación esté en el punto en que esté, o bien puede ir al lugar que desee cambiando la dirección, o también se puede desplazar hacia atrás tanto como desee. Hay algunas directrices para prevenir el comportamiento aleatorio del usuario, que en caso de seguirlas van a ahorrarnos más de un quebradero de cabeza. Veamos algunas de ellas.

TENER EN CUENTA LOS DOBLES ENVÍOS DE FORMULARIOS

En el diseño de nuestro programa, hemos de tener en cuenta que muchas veces el usuario rellena un cierto formulario *HTML* y lo envía, pero seguidamente, no está de acuerdo con algún dato introducido. Entonces, pulsa el botón atrás, modifica el formulario y a continuación lo vuelve a enviar.

LAS TRANSACCIONES DE BASES DE DATOS

Las transacciones que realicemos con una base de datos deberemos intentar cerrarlas una vez se hayan realizado. Si hay dos peticiones diferentes, que usen la misma deberemos cerrar la transacción y cuando se ejecute la segunda volver a abrirla, ya que nada nos asegura que en el momento justo de intersección entre las dos el usuario decida cancelar el proceso. Si dejamos la transacción abierta, los recursos del sistema pueden disminuir de una forma considerable, además de tener una parte de la base de datos bloqueada por la transacción. Deducimos que los demás usuarios no van a poder acceder a esa parte de la base de datos. Una forma de minimizar este riesgo es intentar hacer los cambios en la base de datos al final de la petición en vez de en el principio o en la mitad.

VOLVER A LLENAR ESTRUCTURAS DE DATOS

Si un usuario, cuando se encuentra en la mitad de una aplicación decide dar marcha atrás (por ejemplo para modificar algún dato erróneo), no querrá volver a introducir todos los datos, por lo que deberemos ahorrarle dicho trabajo y será necesario restablecer el valor de origen de las diferentes variables utilizadas.

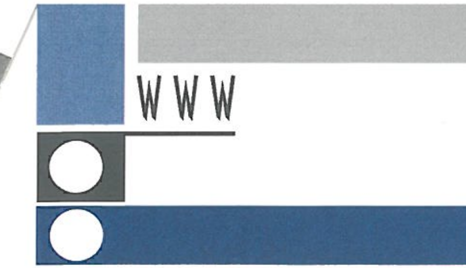
EXPLORACIÓN ABIERTA

Deberemos diseñar la aplicación de tal modo que los usuarios puedan explorar libremente y según la ruta que a ellos les convenga.

LOS ESTADOS EN APLICACIONES IIS

Como ya hemos comentado, normalmente todas las aplicaciones de *Internet* no tienen un estado actual, ya que el protocolo que transmite las peticiones y las respuestas entre el explorador y el servidor *Web* no es capaz de conservar la información entre dos peticiones sucesivas. Para remediar esto utilizamos varios métodos: el objeto *WebClass*, u otros objetos del servidor, aunque aquí nos enfrentamos con el problema de que el servidor necesitará mantener activa la instancia del objeto. O bien nos queda una última solución que consiste en intercambiar información entre el explorador y el servidor de *Web* con *cookies* o con campos *HTML* ocultos, aunque dicha solución es menos segura y aumentará el ancho de banda .

En este artículo hemos visto las bases, y algunos de los problemas con que podemos encontrarnos dentro de las aplicaciones del tipo *IIS*. Ahora ya estamos preparados para empezar a desarrollar un ejemplo práctico que será el fundamento del próximo artículo.



XML. El nuevo estándar de Internet (I)

Adolfo Aladro (aaladro@arrakis.es)

XML se perfila como uno de los estándares que va a despuntar a lo largo de 1999. Gracias a él los navegadores realizarán, tareas de tratamiento de la información evitando así que muchas operaciones requieran el tráfico a través de la red.

INTRODUCCIÓN

En la *WWW* no existe una proporción razonable en lo que respecta al reparto de las tareas informáticas que llevan a cabo los ordenadores. Mientras los servidores hacen el 95% del trabajo, los navegadores sólo efectúan el 5%. Este descompensado balance cobra especial importancia cuando estamos hablando de datos. Cada vez son más frecuentes las páginas *web* que contienen tablas, estadísticas, formularios repletos de datos que proceden de alguna fuente de información estructurada. Todo lo que tenga que ver con el tratamiento y manipulación de los mismos se hace en el lado del servidor, y los datos llegan al usuario en forma de páginas *web*, es decir, sin estructura lógica alguna. Los datos, convertidos en texto *HTML*, pierden toda su capacidad operativa. El usuario ya no puede hacer nada con ellos salvo una lectura lineal de los mismos. Esta situación origina continuas llamadas al servidor, el cual tiene que cargar con la mayor parte de las tareas relacionadas con el tratamiento de la información.

El *W3C* (*World Wide Web Consortium*), organismo encargado de la normalización de estándares para *Internet*, hizo pública la propuesta de estándar del formato *XML* (*eXtensible Markup Language*) el 10 de febrero de 1998. Este estándar nace con el propósito de dar respuesta a todas las cuestiones anteriormente planteadas. El lenguaje *XML* permite incluir metadatos (datos que describen datos) dentro de los documentos *web* de modo que los navegadores pueden realizar, en el ordenador del usuario, tareas de manipulación de la información, eludiendo de esta manera las continuas conexiones al servidor. El *XML* va a hacer posible que tareas como las que acabamos de describir sean habituales dentro de *Internet*.

Todavía existe mucha confusión alrededor de esta nueva tecnología y a veces podemos oír comentarios confusos. En ningún caso *XML* será un sustituto del actual *HTML*. Muy al contrario, lo que se prevé es una utilización complementaria entre ambos para llevar a cabo con éxito aplicaciones *web*.

El lenguaje HTML no es una buena herramienta para representar datos

El lenguaje *HTML* seguirá siendo utilizado para determinar y organizar la visualización de los contenidos, mientras que *XML* se utilizará para dotar a esos contenidos de estructura lógica de forma que puedan ser manipulados como datos. Todos los ordenadores presentes en *Internet* podrán intercambiar información sin tener que preocuparse por las diferencias entre plataformas, arquitecturas o bases de datos. Este es sin duda alguna uno de los grandes avances que promete el lenguaje *XML*.

Finalmente, otras de las propiedades destacadas de este estándar es que *XML* soporta vocabularios especializados. Esto significa que los datos estructurados mediante *XML* pueden adaptarse fácilmente a sectores empresariales tales como compañías de telecomunicaciones, financieras o multimedia. Este tipo de vocabularios específicos se aplicarán a medio plazo en comercio electrónico, gestión de contenidos de *Internet*, integración de aplicaciones y mensajería, etc. De esta manera las diversas empresas podrán implementar nuevos contenidos basados en elementos que ellos definan. A continuación vamos a resumir en algunos puntos las ventajas que aportará la utilización de *XML*:

- Se conseguirá mayor precisión en las búsquedas ya que cuando se utilizan metadatos la efectividad de los motores de búsqueda se incrementa notablemente.
- Se podrá reemplazar en muchos casos los documentos en papel que se intercambian las empresas, como facturas, contratos o pólizas de seguros.
- Facilitará las tareas de clasificación y procesamiento de información, como la identificación de un usuario que realiza operaciones de comercio electrónico, los documentos, etc., que podrán unificarse en un solo estándar. Las páginas generadas con este nuevo estándar son ideales como plataforma de transferencia de datos .
- Permitirá que los datos se encuentren a nivel local para ser utilizados en cálculos. En primer lugar la información será leída por un navegador. Entonces los datos podrán ser manipulados mediante lenguajes de script u otros que soporten el modelo de objetos *XML*.
- Proporcionará al usuario una visión estructurada de la información. Los datos que lleguen a la máquina del cliente podrán visualizarse de muy diversas formas siendo posible personalizar para determinados grupos de usuarios de manera dinámica mediante el establecimiento de preferencias y parámetros de configuración.

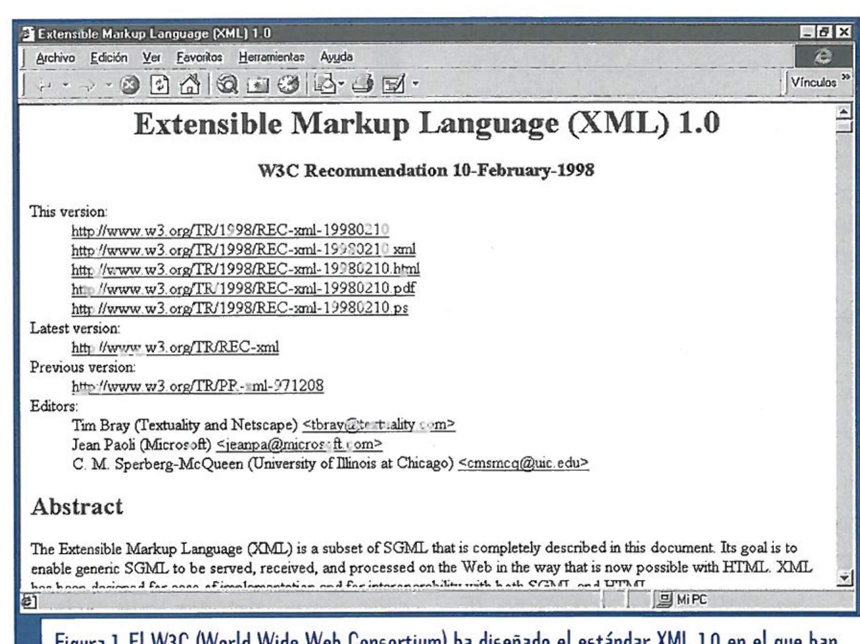


Figura 1. El W3C (World Wide Web Consortium) ha diseñado el estándar XML 1.0 en el que han de basarse todas las implementaciones de XML.

- Será posible integrar datos procedentes de fuentes diversas. Normalmente, dada la naturaleza de la Red, en las aplicaciones *web* suele darse la necesidad de tener que tratar con datos que se originaron en varias fuentes. *XML* facilitará las tareas de compartición, procesamiento y distribución de los mismos ya que una de las cualidades que caracterizan a este estándar es que es extensible y puede utilizarse para describir los datos de una gran cantidad de aplicaciones.

A diferencia del lenguaje HTML las etiquetas XML describen el contenido en vez de presentarlo

- Se mejorará el rendimiento a través de actualizaciones granulares de la información. Los desarrolladores no tendrán que enviar la estructura de los datos cada vez que se produzca un cambio.

Como siempre que se habla de una nueva tecnología es preciso analizar el tipo de soporte que ofrecen las herramientas actuales. Se prevé que 1999 sea el año decisivo para el *XML*. *IBM*, *Microsoft* o *Sun* ya han anunciado su intención de apostar por el siguiente paso tecnológico en la Red. *Software AG* ha anunciado también un producto de bases de datos para el presente año. En cuanto a los navegadores, *Internet Explorer 5.0* da soporte a esta tecnología, como ya veremos a lo largo de esta serie de artículos, y así se prevé que ocurrirá con la versión 5.0 de *Netscape Communicator*.

SOPORTE DE IE 5.0 PARA XML

Internet Explorer 5.0 es el navegador que vamos a utilizar para desarrollar los ejemplos de esta serie de artículos. La razón por lo que se ha hecho esta selección es porque de momento es el primer navegador que soporta, razonablemente bien, el estándar *XML* 1.0 y ya está disponible en su versión prácticamente definitiva. De

todas maneras todos los conocimientos que veamos sobre *XML* se podrán aplicar a cualquier tipo de entorno y/o herramienta *XML*, ya que *Microsoft* se ha basado con bastante fidelidad en el estándar propuesto por el *W3C*. El soporte que proporciona *Internet Explorer 5.0* se puede resumir en:

- Visualización directa de los archivos *XML*. Los documentos *XML* pueden visualizarse en el navegador tal y como se hace con las páginas *HTML*. Además se pueden asociar archivos *CSS* o *XSL* para determinar el aspecto con el que se muestra el documento *XML*.
- Motor de validación de documentos *XML* de alto rendimiento. *IE 5.0* ha mejorado y perfeccionado el *parser* de *XML* que ya empezó a desarrollar en su versión anterior para que soporte de manera más eficiente todos los aspectos del estándar *XML* 1.0.
- Soporte para el lenguaje *XSL* . Este lenguaje sirve para que los desarrolladores puedan aplicar hojas de estilo a los datos *XML*, y proporciona una forma flexible de visualizar dinámicamente los datos.
- Esquemas *XML* (*XML Schemas*). Los esquemas *XML* definen las reglas que debe cumplir un documento *XML*. Podemos decir que son los sucesores de los archivos *DTD*, que como ya veremos a continuación sirven para determinar las reglas gramaticales que debe cumplir un documento de tipo *XML*.
- El modelo de objetos *XML* (*XML DOM*). Gracias al modelo de objetos *XML* podremos acceder mediante *JavaScript* a las propiedades y métodos asociados a un objeto *XML*. Esto permitirá interaccionar dinámicamente dentro de una página *HTML* con los datos proporcionados por *XML*, que podrán estar también presentes en la propia página o en un archivo con la extensión **.XML** aparte.

HERRAMIENTAS XML

En realidad podemos decir que con *XML* pasa algo parecido a lo que ocurre con los archivos *HTML*: basta un simple editor de texto. No obstante, es cierto que a medida que los documentos crecen y las proyectos se hacen más sofisticados, la utilización de herramientas específicas contribuyen a facilitar las labores de edición y mantenimiento. Aunque de momento nosotros sólo vamos a aproximarnos al estándar *XML* a través de ejemplos sencillos, también intentaremos introducir algunas de estas aplicaciones gratuitas que podemos obtener a través de *Internet*.

Se prevé que 1999 sea el año decisivo para el XML

Éstas nos ayudarán a comprender mejor nuestros desarrollos así como a acostumbrarnos a la metodología y entorno de trabajo. En el presente artículo vamos a presentar las siguientes aplicaciones: *XML Notepad*, de *Microsoft*, para la edición de archivos *XML*; y *ezDTD*, para la edición de archivos *DTD*. A continuación veremos cómo son estos archivos pero por el momento nos basta con saber que *XML Notepad* sirve para la edición de cualquier fichero *XML*. Ofrece una interfaz de usuario simple que muestra la estructura en árbol del documento *XML*. *exDTD*, por su lado, hace algo similar con los archivos *DTD*.

¿QUÉ ES XML?

Nosotros ya estamos familiarizados con los lenguajes de etiquetas tales como *HTML*. Por ejemplo, los desarrolladores de aplicaciones *web* suelen utilizar con frecuencia tablas para presentar información acerca de sus productos. Veamos un típico ejemplo.

```
<TABLE BORDER="1">
 <TR>
  <TH>C&oacute;digo</TH>
  <TH>Nombre</TH>
  <TH>Precio</TH>
 </TR>
 <TR>
  <TD>596-AS-78896523</TD>
  <TD>Pentium II 266 MHz</TD>
  <TD>150000</TD>
 </TR>
</TABLE>
```

Se trata de una tabla sencilla donde la primera fila contiene los nombres descriptivos de las columnas, y a continuación se disponen las filas de datos. Si queremos entender claramente el sentido de la tecnología *XML* es indispensable que nos demos cuenta de lo insuficiente que resulta nuestra pequeña tabla. Los datos en ella contenidos no pueden ser tratados como tales.

XML utiliza etiquetas también pero a diferencia del lenguaje *HTML* las etiquetas *XML* describen el contenido en vez de presentarlo. Puesto que se trata de describir los datos y estos pueden ser de muy diversa índole, el lenguaje *XML* permite crear nuestras propias etiquetas así como los atributos de las mismas. La única restricción a la que debemos atender es un conjunto de normas de sintaxis, que como ya veremos más adelante, están hechas con el fin de garantizar la consistencia de los datos representados. El ejemplo anterior podría reescribirse como:

```
<PRODUCTO>
 <CODIGO>596-AS-78896523</CODIGO>
 <NOMBRE>Pentium II 266 MHz</NOM-
     BRE>
 <PRECIO>150000</PRECIO>
</PRODUCTO>
```

Pensemos por un momento en la estructura lógica de nuestros datos. Nosotros queremos realizar tablas de

productos. Un producto estará formado siempre por un código de producto, su nombre y su precio. Pues bien, podemos crear las etiquetas <PRODUCTO>, <CODIGO>, <NOMBRE> y <PRECIO> para representar esas entidades.

Por ahora no debemos preocuparnos por cómo se van a mostrar esos datos en el navegador. Ya veremos más adelante la forma en que se integra el código *XML* dentro de un documento *web*. Ahora solamente tenemos que pensar que de esta manera resulta mucho más fácil buscar un producto determinado en varios servidores o recibir, por ejemplo, una lista de productos personalizada según nuestras preferencias. Lo que antes eran líneas de texto ahora se ha convertido en un esquema estructurado de datos. Solamente necesitamos una aplicación que sea capaz de entender el lenguaje *XML*. Con el ejemplo anterior ya hemos visto dos de las más importantes cualidades del lenguaje *XML*: su sencillez y su carácter extensible. Gracias a las etiquetas *XML* podemos describir los datos utilizando el lenguaje natural, imprimiendo además una estructura jerarquizada.

¿DE DÓNDE PROCEDE XML?

XML es un subconjunto del *SGML* (*Standard Generalized Markup Language*). *SGML* fue estandarizado en 1986 y se basa en el *GML* (*Generalized Markup Language*) inventado por *IBM* en 1969. *XML* fue simplificado con el propósito de ser utilizado dentro de la Red y fundamentalmente para el intercambio de datos.

Las simplificaciones no redunda en perjuicio de la extensibilidad del mismo. Solamente se trata de facilitar la tarea de desarrollar los archivos *XML* válidos.

APLICACIONES DE LA TECNOLOGÍA XML

Ahora que hemos empezado a vislumbrar las posibilidades de la tecnología *XML* es mucho más fácil entender el motivo por el que se prevé que su utilización se extenderá en casi todas las aplicaciones basadas en la Red. Algunas áreas de negocios que se van a beneficiar del uso de *XML* son las siguientes:

COMERCIO ELECTRÓNICO

El comercio electrónico está empezando a despuntar en estos momentos y cada vez es más patente la necesidad de tecnologías que sepan dar un mejor sorpote a este tipo de transacciones. En este sentido pronto podremos contemplar la aparición de vocabularios específicos basados en *XML* que servirán para describir transacciones, órdenes, inventarios y facturas. Estos vocabularios permitirán a los fabricantes, intermediarios y consumidores, incluso a los sistemas bancarios, compartir los mismos datos. En este ámbito la tecnología *XML* jugará un papel indispensable ya que gracias a ella podrán comunicarse e intercambiar datos ordenadores de la Red sin importar los sistemas de información particulares que estos posean. *XML* es un estándar abierto y por esta misma razón se convierte en una elección excelente para todo este tipo de tareas.

Los documentos XML deben tener una estructura de etiquetas jerarquizada

Curiosamente uno de los problemas más importantes hoy en día que plantea *Internet* tiene que ver con una de sus cualidades más relevantes: el inmenso volumen de información que habita en la Red. . Al introducir una palabra para iniciar una búsqueda los motores de búsqueda suelen indagar en aquellas páginas *HTML* que contienen alguna referencia a esa palabra, lo que da como resultado una cantidad de direcciones que en la mayoría de los casos nos conducen a una información que no es la deseada.

Es cierto que cada vez los buscadores disponen de mejores mecanismos de

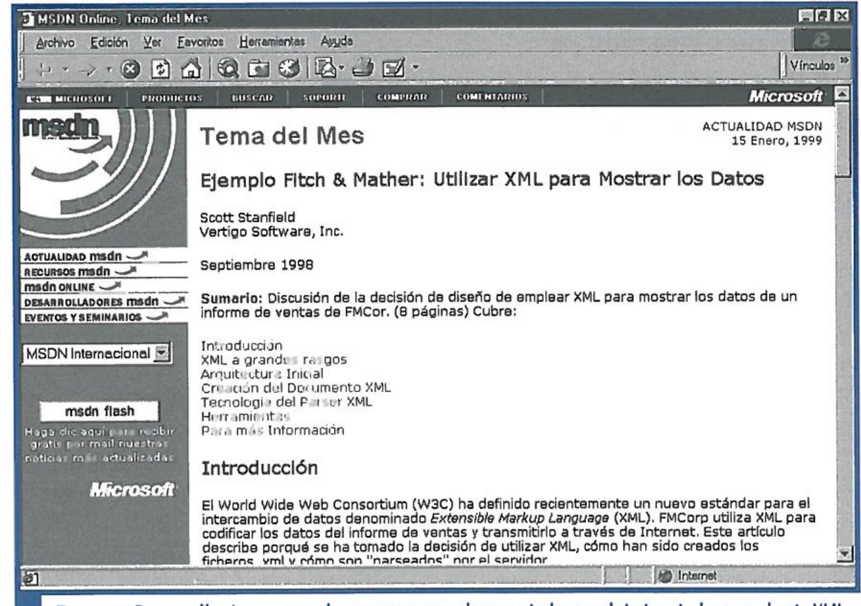


Figura 2. Desarrollo de un ejemplo que muestra algunas de las cualidades de la tecnología XML.

filtrado pero a pesar de todo sus posibilidades son muy limitadas debido la forma en que se encuentra la información en *Internet*. En cambio, si la información dentro de la Red se halla en formato *XML*, esto significará que tendremos una descripción de su contenido y por lo tanto los usuarios podrán encontrar su producto o servicio mucho más rápidamente. Los metadatos ofrecerán un abanico de criterios de búsqueda suficiente como para aumentar notablemente las posibilidades de acertar.

ACCESIBILIDAD DE LOS DATOS

El intercambio de datos en *Internet* forma parte, indudablemente, de su futuro más inmediato y en este sentido la tecnología *XML* viene a cubrir un vacío existente en la actualidad. Debido a la diversidad de plataformas y sistemas de bases de datos hoy en día es muy difícil que se produzca el intercambio de información entre ordenadores remotos. *XML* permitirá especificar los tipos de datos y las relaciones existentes entre ellos, y así los documentos *XML* servirán perfectamente como vehículo de intercambio de información entre distintos sistemas. Incluso se han propuesto otras extensiones del estándar *XML*, como *XQL*, que permitirán realizar consultas tal y como lo hacemos con *SQL*. De este modo la información procedente de diferentes bases de datos será accesible de manera inmediata en forma de documentos *XML* dentro de *intranets* corporativas o en la Red.

SIMPLIFICACIÓN DE LAS APLICACIONES

El consumo de memoria y disco que realizan hoy en día las aplicaciones se ve penalizado por la diversidad de formatos existentes. Los procesadores de texto o las aplicaciones de tratamiento de imágenes a menudo están obligados a leer y escribir en muchos formatos distintos. *XML*, permitirá representar de forma natural la funcionalidad de todos los distintos formatos mediante gramática específicas para cada dominio. Así las aplicaciones solamente tendrán que leer y escribir *XML* teniendo en cuenta la gramática propia del suyo propio. Además, *XML* mejorará los procesos de compartición de datos a través de *Internet*. Estos podrán visualizarse en cualquier navegador de cualquier plataforma.

```
discos.xml - Bloc de notas
Archivo  Edición  Buscar  Ayuda
<?xml version="1.0"?>
<!DOCTYPE DISCOS SYSTEM "discos.dtd">
<DISCOS>
  <DISCO>
    <ARTISTA>Madonna</ARTISTA>
    <TITULO>Ray Of Light</TITULO>
    <ANYO>1994</ANYO>
    <CANCIONES>
      <CANCION>Drowned World/Substitute For Love</CANCION>
      <CANCION>Swin</CANCION>
      <CANCION>Ray Of Light</CANCION>
      <CANCION>Candy Perfume Girl</CANCION>
      <CANCION>Skin</CANCION>
      <CANCION>Nothing Really Matters</CANCION>
      <CANCION>Sky Fits Heaven</CANCION>
      <CANCION>Shanti/Ashtangi</CANCION>
      <CANCION>Frozen</CANCION>
      <CANCION>The Power Of Good-bye</CANCION>
      <CANCION>To Have And Nor To Hold</CANCION>
      <CANCION>Little Star</CANCION>
      <CANCION>Merl Girl</CANCION>
    </CANCIONES>
  </DISCO>
</DISCOS>
```

Figura 3. Para desarrollar archivos XML basta un simple editor de texto como por ejemplo el bloc de notas de Windows.

EL ESTÁNDAR XML

Todas las aplicaciones que implementen *XML* deberán basarse en el estándar propuesto por el *W3C*. Este extenso documento describe todos los elementos y reglas del lenguaje. Su estudio detallado se sale fuera del ámbito de esta serie de artículos, aunque recomendamos al lector interesado que lo consulte.

Los valores de los atributos deben escribirse obligatoriamente entre comillas (simples o dobles)

Nosotros lo que haremos será resumir desde una óptica práctica todos aquellos aspectos más importantes e indispensables para empezar a trabajar con *XML*.

ESTRUCTURA JERÁRQUICA DE LOS ELEMENTOS

Los documentos *XML* deben tener una estructura de etiquetas jerarquizada. Esto es, todas las etiquetas iniciales deben tener su correspondiente etiqueta final. Dentro de *XML*, un par de etiquetas (inicial y final) se denomina un elemento. Cualquier elemento debe estar adecuadamente anidado dentro de otro. Veamos un ejemplo :

<AUTOR><NOMBRE></AUTOR>
</NOMBRE>

XML resulta muy riguroso en este punto: o el elemento **AUTOR** está contenido dentro del elemento **NOMBRE**, o viceversa, pero nunca la situa-

ción anterior, u otra parecida, en la que no es posible determinar una estructura clara de la información descrita. Nótese que el lenguaje *HTML* no es tan estricto a la hora de considerar que un documento está bien estructurado.

```
<B>Sólo negrita...<I>Negrita y
    cursiva...</B></I>
```

ETIQUETAS VACÍAS

XML permite incluir etiquetas vacías, es decir, que son en sí mismas inicial y final y no contienen nada. Éstas se identifican por terminar con el carácter /.

```
<FECHA DIA="24" MES="7" ANYO="1999"
    />
```

El lenguaje *HTML* también permite este tipo de etiquetas pero no es necesaria la aparición del carácter / al final de las mismas. Un ejemplo de esto es la etiqueta *
*:

```
Esto es un texto...<BR>Esto es después del
    salto de línea
```

La etiqueta *
* si la trasladásemos tal cual al lenguaje *XML* produciría errores ya que al no llevar la barra / el *parser* interpretaría que es la inicial,

ELEMENTOS RAÍZ ÚNICOS

Los documentos *XML* solamente permiten un elemento raíz. Esta restricción sirve para verificar que el documento está completo. Así por ejemplo, si tenemos información de varias personas:

```
<PERSONA>
  <NOMBRE>Fernando</NOMBRE>
</PERSONA>
<PERSONA>
  <NOMBRE>Pepe</NOMBRE>
</PERSONA>
...
```

la sintaxis *XML* nos obliga a agruparlas dentro de una solo elemento:

```
<AMIGOS>
  <PERSONA>
    <NOMBRE>Fernando</NOMBRE>
  </PERSONA>
  <PERSONA>
    <NOMBRE>Pepe</NOMBRE>
  </PERSONA>
  ...
</AMIGOS>
```

VALORES DE ATRIBUTOS ENTRE COMILLAS

Las etiquetas *XML* pueden tener atributos, que al igual que ocurre con las propias etiquetas, son definidos por nosotros mismos. Los valores de los atributos deben escribirse entre comillas (simples o dobles).

Un documento XML es válido si cumple las reglas de su DTD

El lenguaje *HTML* permiten atributos sin entrecomillar. Así podemos encontrarnos fácilmente en una página *HTML* la siguiente situación:

```
<IMG SRC="img01.gif" WIDTH=23
    HEIGHT=98>
```

Nótese que el documento con la etiqueta anterior en el lenguaje *XML* se consideraría mal formado por dos razones: falta el carácter / que determina que esa es una etiqueta sin contenido (es en sí misma final e inicial), y los valores asignados a los atributos *WIDTH* y *HEIGHT* no están entre comillas.

DISTINCIÓN ENTRE MAYÚSCULAS Y MINÚSCULAS

Las etiquetas *XML* distinguen entre mayúsculas y minúsculas, cosa que no ocurre con el lenguaje *HTML*. Podemos escribir las dos siguientes líneas de código dentro de una página *web* y obtendríamos el mismo efecto:

```
<IMG SRC="img01.gif" WIDTH=23
    HEIGHT=98>

<img src="img01.gif" width=23 height=98>
```

```
Sin embargo de cara al lenguaje XML ocurrirían
    errores si se dieran situaciones como la
    siguiente:
```

```
<PERSONA>
  <nombre>Fernando</NOMBRE>
</PERSONA>
```

Por este motivo resulta recomendable acostumbrarse a escribir los archivos *XML* desde el principio todo en mayúscula o todo en minúscula.

ESPACIOS EN BLANCO

Los espacios en blanco dentro de las etiquetas, entre los datos, son irrelevantes. Esto quiere decir que entre estas dos etiquetas no hay ninguna diferencia:

```
<FECHA DIA="24"     MES="7"
```

```
        ANYO="1999" />
<FECHA DIA="24" MES="7" ANYO="1999"
        />
```

Dentro de las propias etiquetas, entre las comillas que determinan los valores de los atributos, los espacios en blanco se eliminan o normalizan.

```
FECHA DIA="24   " MES="7"
        ANYO="1999" />
<FECHA DIA="24" MES="7" ANYO="1999"
        />
```

Esto último es muy importante tenerlo en cuenta ya que puede inducir fácilmente a errores. Las reglas de normalización son bastante complejas con lo que conviene referirse al estándar si queremos profundizar en ellas.

CODIFICACIÓN DE LOS CARACTERES

XML permite especificar diferentes formatos de codificación de caracteres. Estos deben de estar especificados dentro de la etiqueta <?*XML* ?>:

```
encoding=="UTF-8".
```

CARACTERES ESPECIALES

Hay varios caracteres que forman parte de la estructura sintáctica de *XML* y no son interpretados como simples caracteres si se sitúan dentro del documento por lo que deben ser sustituidos por ciertas secuencias de caracteres denominadas entidades predefinidas. Algo similar ocurre en el lenguaje *HTML* con ciertos caracteres. Por ejemplo, la letra "a" acentuada, en *HTML* la representamos como **"á"**. En *XML* se dan situaciones parecidas. La Tabla 1 muestra algunos caracteres reservados y sus secuencias.

DTD (DOCUMENT TYPE DECLARATION)

Probablemente la diferencia más notable entre *HTML* y *XML* es que con este último es posible crear nuevas etiquetas que se adapten a las necesidades de nuestros datos. Ahora bien, si creamos nuevas etiquetas, en la mayoría de los casos nos interesará definir o restringir su uso y sintaxis mediante reglas gramaticales que las etiquetas habrán de obedecer. Esas reglas gramaticales vendrán dadas en forma de documentos *DTD* (*Document Type Declaration*):

- Qué etiquetas se permiten dentro de qué etiquetas.
- Qué etiquetas y/o atributos son opcionales.

En cuanto a los *DTD*, un documento *XML* puede hacer una de la siguientes cosas:

- Referirse a un *DTD* utilizando una dirección *URL*.
- Incluir un *DTD* como si fuera una parte del documento *XML*.
- Omitir la presencia de un *DTD*. Si no existe un *DTD* puede verificarse que un documento *XML* esté bien o o esté mal formado, pero nada se puede decir acerca de su validez.

VALIDO VS. BIEN FORMADO

Un documento *XML* es válido si su contenido cumple las reglas de su *DTD* asociado. Esta cualidad - la validez de un documento *XML* - sirve para que las aplicaciones puedan estar seguras de que los datos *XML* están completos, correctamente formateados y además los valores de sus atributos son los adecuados. Cuando un documento *XML* no hace referencia a ningún *DTD* no podemos decir nada acerca de su validez. En esos casos el *parser* solamente puede informarnos acerca de si el documento está bien formado (es decir, respeta la sintaxis de lenguaje *XML*).

SINTAXIS DE UN ARCHIVO DTD

Un archivo *DTD* representa una gramática que describe qué etiquetas y atributos son válidos dentro de un documento *XML*, y en qué contexto son válidos. Normalmente las normas y reglas que debemos seguir para construir un archivo *DTD* asociado a un documento *XML* suelen ser muy fáciles. Veámoslo a través de un ejemplo. Supongamos que partimos de un documento *XML* que pretende guardar los datos relativos a nuestra colección de discos. Este archivo podría presentar la siguiente forma:

```
<?XML version="1.0"?>
<DISCOS>
```

Tabla 1. Algunos caracteres reservados y sus secuencias sustitutivas.

Carácter reservado	Secuencia sustitutiva
<	<
&	&
>	>
"	"
'	'

```
<DISCO>
  <ARTISTA>Madonna</ARTISTA>
  <TITULO>Ray Of Light</TITULO>
  <ANYO>1994</ANYO>
  <CANCIONES>
    <CANCION>Drowned World/Substitute
      For Love</CANCION>
    ...
    <CANCION>Merl Girl</CANCION>
  </CANCIONES>
 </DISCO>
</DISCOS>
```

En primer lugar todos los datos se agrupan en torno al elemento definido por la etiqueta *<DISCOS>*. Como cabe esperar, cada elemento definido por la etiqueta *<DISCO>* contendrá toda la información relativa a un disco de nuestra colección: el nombre del artista (*<ARTISTA>*), el título del disco (*<TITULO>*), el año de la producción (*<ANYO>*) y una lista con todas las canciones del disco (*<CANCIONES>* y *<CANCION>*).

Un archivo DTD representa una gramática que describe qué etiquetas y atributos son válidos

Un *DTD* es simplemente un archivo de texto que pretende describir de un modo formal todo lo que acabamos de decir. Vayamos paso a paso. La primera línea de nuestro archivo *DTD* sería:

```
<!ELEMENT DISCOS (DISCO)+>
```

Cada línea de un *DTD* obedece a la siguiente estructura:

Tipo	Declaración del elemento	Modelo de contenido del elemento
ELEMENT	DISCOS	(DISCO)

Ahora debemos fijarnos en el modelo de contenido asociado al elemento **DISCOS**. Éste nos informa, como su propio nombre indica, de cual es la estructura del contenido asociado a ese elemento. En primer lugar los paréntesis solamente sirven para agrupar (en este caso sólo agrupan a un elemento, pero es igual). El símbolo + sirve para indicar que dentro del elemento **DISCOS** se van a hallar una o más ocurrencias del elemento **DISCO**, lo que dicho en lenguaje natural significa que una colección de discos tendrá un disco como mínimo y un número indeterminado como máximo. La Tabla 2 muestra algunas de las normas más importantes a la hora de redactar un documento *DTD*. Vemos como se pueden definir distintos modelos de contenido de elementos dependiendo de cada situación. Continuemos con nuestro *DTD*. La segunda línea del mismo sería:

```
<!ELEMENT DISCO
    (ARTISTA,TITULO,ANYO,CANCIO-
    NES)>
```

Esto significa simplemente que un elemento **DISCO** está formado por cuatro elementos más a su vez: **ARTISTA**, **TITULO**, **ANYO** y **CANCIONES**. Los paréntesis agrupan los elementos y éstos se separan por comas. El resto de nuestro *DTD* quedará como sigue:

```
<!ELEMENT ARTISTA (#PCDATA)>
<!ELEMENT TITULO (#PCDATA)>
<!ELEMENT ANYO (#PCDATA)>
<!ELEMENT CANCIONES (CANCION)+>
<!ELEMENT CANCION (#PCDATA)>
```

Los elementos **ARTISTA**, **TITULO** y **ANYO** ya no contienen en sí mismos estructuras más complejas. La palabra reservada *#PCDATA* sirve para indicar que se trata de un dato en forma de cadena de texto codificada según el esquema de codificación que se indique en el documento *XML*. Por último, cada disco podrá tener una o más canciones y por cada canción simplemente guardamos su título.

LOS ATRIBUTOS

Como ya hemos apuntado las etiquetas definidas por nosotros mismos pueden llevar atributos. Pues bien, estos atributos también pueden estar sujetos a normas y éstas se especifican también a través de un documento *DTD*. Supongamos que tenemos el siguiente documento *XML*:

```
<?XML version="1.0"?>
<AGENDA>
 <PERSONA GENERO="masculino">
```

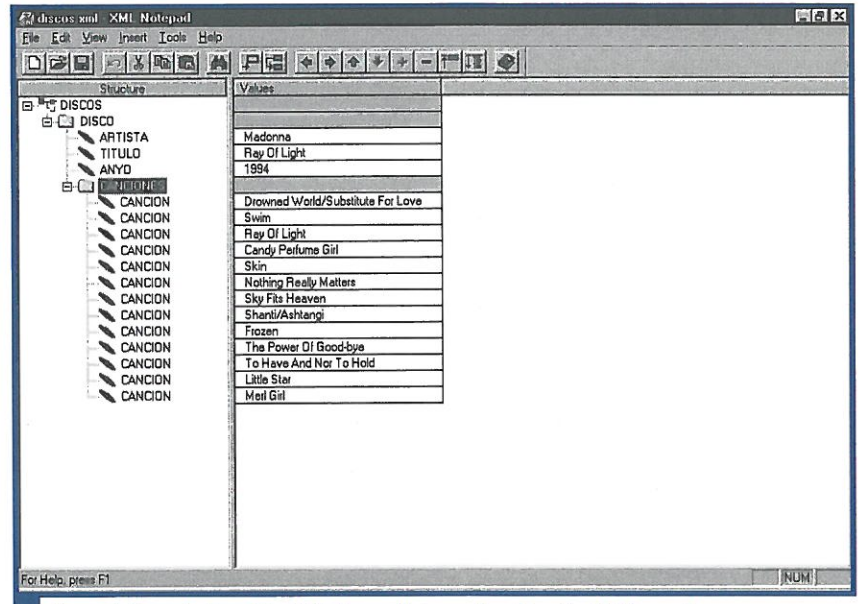


Figura 4. Ejemplo de archivo XML desarrollado con la aplicación XMLNotepad.

```
  <NOMBRE>Adolfo</NOMBRE>
  <EMAIL>aaladro@arrakis.es</NOMBRE>
 </PERSONA>
</AGENDA>
```

Hemos considerado el atributo *GENERO* en el elemento **PERSONA**. En nuestro *DTD* este atributo podría quedar reflejado de la siguiente manera:

```
<!ATTLIST PERSONA GENERO
      (masculino|femenino) #IMPLIED >
```

La línea anterior obedece a la siguiente estructura:

Tipo	Elemento
<!ATTLIST	PERSONA
Atributo	Definición del atributo
GENERO	(masculino|femenino)
	#IMPLIED >

La expresión *(masculino|femenino)* es un tipo enumerado que indica que el atributo *GENERO* puede tomar uno de esos dos valores. La palabra reservada *#IMPLIED* sirve para indicar que el atributo es opcional.

Si no indicamos ningún DTD el validador comprueba la formación del documento XML

Supongamos ahora que quisiéramos recoger la siguiente situación: el género de esa persona puede ser femenino, masculino o desconocido (caso éste que puede darse si el nombre se presta a equívocos y desconocemos el género). Deseamos que el atributo siga siendo opcional pero en el caso de que no se defina queremos que tome el valor por defecto correspondiente a la situación de desconocido. Toda esta premisa podría reflejarse en un *DTD* a través de la siguiente línea:

```
<!ATTLIST PERSONA GENERO
      (masculino|femenino|desconodio)
      "desconocido">
```

O si quisiéramos que el atributo fuese obligatorio utilizaríamos la palabra reservada *#REQUIRED*:

```
<!ATTLIST PERSONA GENERO
      (masculino|femenino|desconodicido)
      #REQUIRED>
```

Otra opción sería que siempre que un atributo esté presente tome un valor predeterminado. Este se podría lograr mediante la palabra reservada *#FIXED*.

TIPOS DE DATOS

En el ejemplo anterior hemos visto el tipo enumerado que consiste en una lista de posibles valores que puede tomar el atributo en cuestión. Los valores están separados por el carácter | y todos se encuentran encerrados entre paréntesis. Los otros dos tipos existentes son: *CDATA* y *tokenized*. El tipo *CDATA* se utilizará para referirnos a cadenas de caracteres. Imaginemos que estamos describiendo mediante un *DTD* el atributo *METHOD* de la etiqueta *<FORM>* del lenguaje *HTML*:

```
<!ATTLIST FORM METHOD CDATA
      #FIXED "POST">
```

Con la línea anterior estaríamos indicando que el atributo *METHOD*, siempre que esté presente, contendrá la cadena de caracteres *POST*.

El tipo *tokenized* se utilizará para representar un conjunto fijo de palabras reservadas con significado especial. A veces es necesario tener un atributo en una etiqueta que sirva para identificar, permitiéndonos así el poder distinguir entre unas y otras. En estos casos utilizamos el atributo *tokenized*.

```
<!ATTLIST PERSONA IDENTIFICADOR ID
      #REQUIRED>
```

La línea anterior, al utilizar la palabras reservadas *ID* y *#REQUIRED* fuerza a todas las etiquetas de tipo **PERSONA** a que tengan el atributo *INDENTIFICADOR*. Su valor será único dentro de todo el documento *XML*.

Si queremos profundizar en todos aspectos del estándar es conveniente que acudamos a la especificación. Más adelante en esta serie de artículos veremos que los archivos *DTD* no resultan la única forma de dotar de una serie de normas y reglas a nuestros datos.

VALIDADOR DE DOCUMENTOS XML

Es el momento de tomar contacto con *XML* de una manera práctica. Para empezar vamos a realizar un validador de documentos *XML* aprovechándonos de las ventajas que ofrece *Internet Explorer 5.0*. Veremos cómo insertar código *XML* dentro de un documento *web* y cómo es posible acceder, manipular y/o verificar esos datos mediante *JavaScript* a través de *DOM* para *XML* que es implementado por el propio navegador. No se trata de hacer un repaso exhaustivo de *DOM* con todas sus propiedades y métodos. Nuestra intención es adentrarnos de forma sencilla en los aspectos prácticos del estándar para luego ir profundizando a lo largo de los sucesivos artículos.

El validador de documentos será una sencilla página *HTML* que contenga un área de texto donde el usuario pueda introducir el código *XML*, y una zona de información donde aparezca el resultado de analizar el código *XML* introducido. La verificación de los datos se hará dinámicamente a medida que se tecleen en el área de texto. En el caso de que el *parser* informe de algún error, se indicará en la zona destinada dentro de la página a mostrar los resultados, señalando la línea y posición donde se

ha producido, así como sus posibles motivos. Si no se producen errores, el icono indicativo señalará que todo es correcto mediante el símbolo :).

El Listado 1 muestra el cuerpo de la página *HTML*. Tenemos un campo de tipo *TEXTAREA* de tamaño **70x15** al que le hemos asignado un determinado estilo mediante el atributo *STYLE* y cuyo identificador será **CodigoXML**. Controlamos dinámicamente la actualización del *TEXTAREA* capturando el evento *keypress*. Así, cada vez que se produce una pulsación dentro del *TEXTAREA* llamamos a la función *verificar*.

El área de información de nuestro validador estará formada por tres capas: *icono*, *mensajeerror* y *dondeerror*. Estas capas, hechas con la etiqueta *<DIV>*, se actualizarán dinámicamente con el resultado que devuelva el *parser*.

La verificación de los datos se hará dinámicamente según se tecleen en el área de texto

Por último, y lo más importante, está la etiqueta *<XML>*. Como es evidente, se trata de una nueva etiqueta cuyo fin es dar la posibilidad de introducir código *XML* dentro de un documento *web*. Por el momento sólo nos interesa saber que podemos asignarle un identificador mediante el atributo *ID* y que éste nos servirá para acceder desde *JavaScript* al objeto correspondiente. También podemos capturar el evento *readystatechange*, el cual informará acerca del estado del proceso de carga de los datos. Los datos *XML* que se introducen en las página *HTML* de esta forma constituyen lo que se denomina como *data island* ("isla de datos").

En principio nuestra isla de datos no contendrá nada. A medida que el usuario teclee en el área de texto cargaremos los datos correspondientes, lo que activará de manera automática el parser de *XML* del navegador. Comprobando el estado resultante de cargar los datos sabremos si el código *XML* introducido hasta el momento es válido y/o está bien formado. La función cargaTerminada, que hemos asignado al evento *readystatechange* se ejecutará cuando la carga se haya terminado, momento en el cual podremos observar el resultado dado por el *parser*.

Listado 1. Código HTML del cuerpo de la página.

```
<B>Validador de documentos XML</B><BR> <BR>
<TEXTAREA NAME="CodigoXML" COLS="70" ROWS="15" STYLE="font-family:
Courier New; font-size: 9pt;" onkeyup="verificarBienFormado()" onfocus="verificar-
BienFormado()"></TEXTAREA>
<BR><A HREF=http://www.towercom.es>S&oacute;lo Programadores</A><BR>
<BR>
<DIV ID="icono" CLASS="icono">:l</DIV>
<DIV ID="mensajeerror" CLASS="mensajeerror"></DIV>
<DIV CLASS="dondeerror" ID="dondeerror"></DIV>
<XML ID="xmldoc" onreadystatechange="cargaTerminada()"></XML>
```

En primer lugar *verificar* toma los datos que hay en el área de texto y utiliza el método *loadXML* del objeto asociado a la isla de datos -a partir de ahora "el objeto *XML*"- para cargar los datos correspondientes. Cuando los datos hayan sido analizados por completo por el navegador se producirá el evento *readystatechange* y por lo tanto la función *cargaTerminada* se ejecutará. Una de las propiedades del objeto *XML* es el objeto **parserError**. Consultando, por ejemplo, el valor de la propiedad *reason* podremos saber si se ha producido algún error. Si esta cadena esta vacía la carga se ha efectuado sin problema alguno y por lo tanto el código *XML* es válido y/o está bien formado.

Para comprobar el funcionamiento de nuestro validador de documentos *XML* vamos a utilizar el ejemplo anterior: la colección de discos. Como se puede observar el nos muestra que el código introducido está en primer lugar bien formado, es decir, sigue el estándar *XML*, y que además es válido, lo que significa que se ajusta al *DTD* asociado. Nótese que si no indicáramos ningún *DTD* el validador lo único que podría comprobar es que el documento *XML* esté bien formado. Ahora si eliminamos una de las líneas del código *XML*, por ejemplo, la que contiene el año de lanzamiento del disco, el validador informará de que el documento no es válido ya que no se ajusta al *DTD* asociado.

Los aspectos de presentación son también muy sencillos. Por un lado utilizamos hojas de estilo para determinar las propiedades de visualización de los distintos elementos de la página (ya sea declarando los estilos al principio de la página mediante la etiqueta *<STYLE>* y luego utilizando el atributo *CLASS* de los elementos, o bien directamente utilizando el atributo *STYLE*). Las propiedades *innerHTML* e *innerText* de los elementos permitirán cambiar dinámicamente su contenido. Asimismo, accediendo a la propiedad *style* podremos cambiar también aspectos como el color de fondo del icono de indicación .

A lo largo de este artículo nos hemos introducido en la tecnología *XML*. Hemos analizado las ventajas que su uso aportará al desarrollo de aplicaciones *web* y las razones por las que es está destinado a asentarse como estándar dentro de las tecnologías *Internet*. En los artículos venideros profundizaremos en estos aspectos y veremos las posibilidades que ofrece el lenguaje *XML*.

Glide (V)

Constantino Sánchez Ballesteros (constantino@nexo.es)

Siguiendo nuestro avance por las interioridades técnicas de *Glide*, ha llegado el momento de abordar los conceptos del mapeado de texturas y los filtros que pueden aplicarse para mejorar el aspecto visual de las escenas creadas.

TEXTURE MAPPING

Hace unos años, se discutía la forma de rellenar un polígono bien mediante un color sólido o con un sombreado mediante la mezcla de varios colores. En nuestro caso abordaremos esta cuestión desde una perspectiva más avanzada; el relleno de polígonos con texturas.

El *Texture mapping* es una técnica en la cual una imagen bidimensional es pegada como papel de empapelar sobre una superficie tridimensional. Esto genera imágenes muy realístas sin requerir el uso de polígonos pequeños detallados.

MAPEADO DE TEXTURAS

Un mapa de texturas es una matriz cuadrada o rectangular que contiene elementos de la textura. La *TMU* contiene memoria para almacenar texturas, circuitería para convertir *texels* en *pixels* y otros circuitos electrónicos para agregar, escalar y mezclar esos *texels*. El subsistema *Voodoo Graphics* incluye al menos una *TMU*, aunque puede tener tres. Cada *TMU* producirá un color *RGBA* de su propia textura almacenada que será combinada para producir un color de textura *RGBA* que pueda ser seleccionado como la entrada a la unidad de combinación de color y combinación alpha.

TEXTURAS Y TEXELS

Como ya se ha dicho, las texturas son matrices de datos cuadradas o rectangulares; cada valor individual de la textura es llamado *texel* y tiene un direccionamiento s, t. Las coordenadas s y t del *texel* entran dentro del rango **[–32768..32767]** y deben ser divididos por w antes de guardarlos en una estructura *GrVertex*. El rango más largo para s y t permite que una textura sea repetida múltiples veces en el polígono que se va a rellenar. Con frecuencia, las texturas mapeadas en los polígonos tienen un valor w diferente a las texturas no proyectadas. Las texturas proyectadas iteran q/w donde w es la distancia homogénea desde el ojo del observador y q es la distancia homogénea de la fuente de origen proyectada.

```
typedef struct {
    float oow;
    float sow;
      /* coordenada de textura s/w */
    float tow;
      /* coordenada de textura t/w */
} GrTmuVertex;

typedef struct {
    float x, y, z;      /* x, y, z del espacio de
    la pantalla. z ignorado */
    float ooz;
    float oow;
      /* 1/w (usado para w buffering) */
    float r, g, b, a;   /* rojo, verde, azul y
    alpha ([0..255.0]) */
    GrTmuVertex
    tmuvtx[GLIDE_NUM_TMU];
} GrVertex;
```

Por defecto, *Glide* asume que todas las coordenadas *w* (*oow*) en la estructura *GrVertex* son idénticas y que todas las coordenadas *s* y *t* también lo son. Si los valores no son idénticos, nuestra aplicación debe alertar a *Glide* para que sepa que los valores en la estructura *GrVertex* son diferentes y que los gradientes necesitan ser calculados de nuevo.

```
void grHints( GrHints_t hintType,
        FxU32 hintMask )
```

Cada *hintType* controla una optimización diferente o modo de operación. El tipo *GR_HINT _STWHINT* controla el parámetro de optimización *stw* y especifica la fuente origen para los valores de esos parámetros. Existe un orden implícito de *TMUs* en *Glide*, comenzando con *TMU0*, seguido de *TMU1* y *TMU2*. En cuanto se lean las coordenadas *s* y *t*, serán transmitidas a las siguientes *TMUs*. Una vez que los valores *sow* y *tow* se hayan leído, no se volverán a leer a menos que se especifique una indicación. Estas indicaciones son utilizadas para ayudar a *Glide* a encontrar las coordenadas *w*. Si se especifica una indicación de *w* y está activada una tabla de niebla, la estructura *tmuvtx[].oow* correspondiente a la *TMU* mencionada en *hintMask* se leerá y enviará a las siguientes *TMUs*.

SISTEMA DE COORDENADAS DE TEXELS

Todos los mapas de texturas cuadrados tienen su origen en (*s,t*) = **(0,0)** y su esquina opuesta en **(256,256)**. Esto es así para mapas de texturas 1x1. Debemos saber que estas coordenadas de texturas son divididas previamente por *w*. La coordenada de textura **(0.5, 0.5)** representa el centro exacto del primer *texel* en un mapa de texturas de **256x256** y **(255.5, 255.5)** representa el centro exacto del *texel* en la esquina opuesta. **(256.5, 256.5)** envolverá desde el centro al primer *texel*. En general, el centro del primer *texel* en un mapa de textura de $2^n x 2^n$ (donde $0 \leq n \leq 8$) se situará en la posición $(128/2^n, 128/2^n)$.

La coordenada w es leída a partir de la estructura GrVertex y transmitida a todas las TMUs

Las texturas rectangulares también tienen su origen en **(0, 0)**. Si la textura rectangular es más ancha que alta (*s* es mayor que *t*), la esquina opuesta se situará en la coordenada **(256, n)** donde *n /256= t/s*. Por ejemplo, si la textura es cuatro veces más ancha que alta, *n*=**64**. De igual modo, si la textura rectangular es más alta que ancha, la esquina opuesta se situará en **(n, 256) y n/256=***s/t*.

MAPEANDO TEXELS

Las texturas mostradas abajo tienen todas un *ratio* de aspecto de **1:2** y un rango en tamaño de 32x64 a 1x2. En cada una, las coordenadas de textura *(s,t)* varían en un rango de **(0,0)** a **(128,256)**. De este modo, los *texels* se hacen más grandes (en términos de cubrir el espacio de coordenadas) cuando el tamaño del mapa de texturas decrementa. Todo el sistema de mapeado de texturas de las *Voodoo* se gestiona en la *TMU*, incluyendo soporte para perspectiva real de mapeado de texturas, *mipmapping* con nivel de detalle *per-pixel* (*LOD*), y filtro bilineal.



TEXTURE FILTERING

Como ya sabemos, los mapas de texturas son cuadrados, pero después de ser mapeados a un polígono y transformados en coordenadas de pantalla, rara vez corresponden los *texels* individuales con los *pixels* del mapa de texturas. Dependiendo de la transformación utilizada y el mapa de texturas aplicado, un simple *pixel* de la pantalla puede aparecer magnificado como una serie repetitiva del mismo *pixel* formando grandes cuadrados en pantalla que restan calidad visual a la representación. Para paliar este problema, *Glide* utiliza diversos tipos de filtros que suavizan el mapeado de texturas aplicado en las escenas *3D* también permite utilizar dos tipos de filtros *Point Sampling* e Interpolación bilineal.

POINT SAMPLING Y FILTRO BILINEAL

Glide soporta dos métodos para elegir un *texel* con un mapa de texturas. Si el *pixel* mapea menos de un *texel*, tal y como se muestra en el diagrama (a), se utilizará una ampliación de texturas. Si el *pixel* mapea a más de un texel, tal y como se muestra en el diagrama (b), se utilizará una reducción. El usuario puede seleccionar entre *point-sampling* y *bilinear filtering* durante el ampliado o reducción. Cuando se utiliza *point sampling*, el *texel* cuyas coordenadas *(s, t)* estén más cerca del centro del *pixel* será el elegido. *Bilinear filtering* computa una media de una matriz 2x2 para los *texels* que estén más cerca del centro del *píxel*. Los filtros

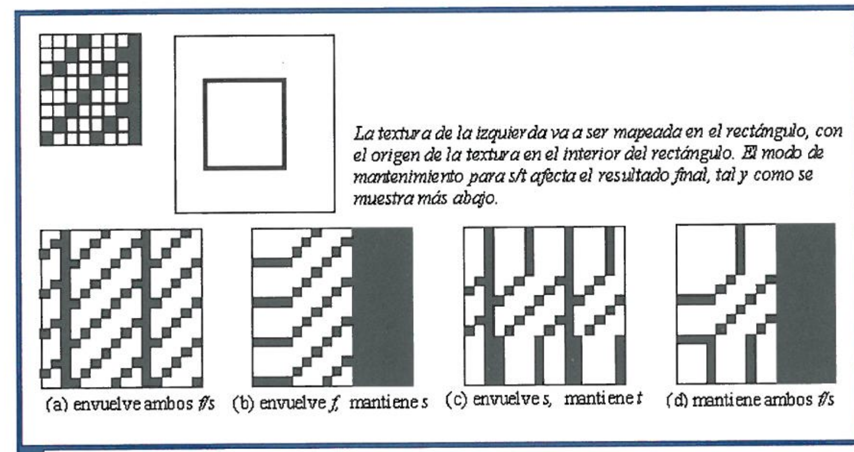


Figura 3. Funcionamiento de Texture Clamping.

El primer paso consiste en incluir las librerías necesarias para poder compilar el programa. Existe una librería especial llamada **TLIB.h** que se debe referenciar indicando su localización en el disco duro.

```
#include <stdlib.h>
...
#include "c:\Archivos de
      programa\Glide\Glide\Src\Sst1\Glide\tests
      \tlib.h"
GrHwConfiguration hwconfig;
static char version[80];
...
```

Definimos diversas estructuras para la carga de archivos gráficos y paletas:

```
typedef FxU32 Palette[256];
typedef struct {
 FxU8  yRGB[16];
 FxI16 iRGB[4][3];
 FxI16 qRGB[4][3];
 FxU32 packed_data[12];
} NCCTable;
...
```

El procedimiento *LoadTexture* se encargará de cargar la textura que utilizar en el ejemplo. Al final del listado podéis encontrar sus rutinas

```
static int loadTexture(const char *filename,
              GrTexInfo *info,
              GrTexTable_t *tableType,
                   void *table );

static GrTexTable_t texTableType( GrTexture-
      Format_t format );
```

Establecemos una resolución de pantalla de 640x480 y formato de color *RGBA* (*Alpha RGB*). Para ello, nos apoyaremos en la función *grSstWinOpen*:

```
void main( int argc, char **argv) {
  char match;
  char **remArgs;
  int  rv;

  GrScreenResolution_t resolution =
      GR_RESOLUTION_640x480;
...

  grGlideInit();
  assert(grSstQueryHardware( &hwconfig ) );
  grSstSelect( 0 );
  assert(grSstWinOpen( 0,
           resolution,
           GR_REFRESH_60Hz,
           GR_COLORFORMAT_ABGR,
           GR_ORIGIN_UPPER_LEFT,
           2, 1 ) );

  tlConSet( 0.0f, 0.0f, 1.0f, 0.5f,
       60, 15, 0xffffff );
```

En este punto del programa estableceremos los filtros y efectos que se van a utilizar. Activaremos el estado del *Render*, textura tipo *Decal* y desactivamos el *mipmapping*. Las funciones de ayuda : *grColorCombine* y *grTexCombine*:

```
grColorCombine
(GR_COMBINE_FUNCTION_SCALE_OTHER,
    GR_COMBINE_FACTOR_ONE,
    GR_COMBINE_LOCAL_NONE,
    GR_COMBINE_OTHER_TEXTURE,
    FXFALSE );
```

Para la unidad de combinación de texturas y *mipmap* utilizaremos *TMU0*.

```
grTexCombine(GR_TMU0,

    GR_COMBINE_FUNCTION_LOCAL,
        GR_COMBINE_FACTOR_NONE,

    GR_COMBINE_FUNCTION_LOCAL,
        GR_COMBINE_FACTOR_NONE,
        FXFALSE, FXFALSE );

grTexMipmapMode( GR_TMU0,
    GR_MIPMAP_DISABLE, FXFALSE );
```

Las texturas pueden representar escenas fotorrealistas con ayuda de los filtros

Cargamos los datos de la textura dentro de la memoria *RAM*. El archivo que debemos cargar tiene que estar en el mismo directorio que el ejecutable. El nombre del fichero gráfico es **miro.3df:**

```
assert( loadTexture("miro.3df",
          &texture.info,
          &texture.tableType,
          &texture.tableData ) );
```

Descargamos los datos de la textura en la *TMU* mediante *grTexDownloadMipmap*:

```
grTexDownloadMipmap(GR_TMU0,
          grTexMinAddress( GR_TMU0
  ),
```

```
GR_MIPMAPLEVELMASK_BOTH,
            &texture.info );

  if ( texture.tableType != NO_TABLE ) {
    grTexDownloadTable( GR_TMU0,
              texture.tableType,
              &texture.tableData );
  }
```

Seleccionamos la textura como fuente de todas las operaciones que trabajen con texturas.

```
  grTexSource(GR_TMU0,
       grTexMinAddress( GR_TMU0 ),
       GR_MIPMAPLEVELMASK_BOTH,
       &texture.info );
```

Creamos los dos triángulos que conformarán el rectángulo que contendrá la textura. Se crea una estructura de vértices. Las variables serán **vtxA**, **vtxB** y **vtxC**.

```
    grBufferClear( 0, 0, GR_ZDEPTHVA-
     LUE_FARTHEST );

    /*——
     A-B
     |\|
     C-D
     ——*/
    vtxA.oow = 1.0f;
    vtxB = vtxC = vtxD = vtxA;
```

Ampliamos o reducimos el rectángulo dependiendo de la selección por parte del usuario:

```
    if (minify) {
      vtxA.x = vtxC.x = tlScaleX( 0.0f );
      vtxB.x = vtxD.x = tlScaleX( 1.0f );
      vtxA.y = vtxB.y = tlScaleY( 0.0f );
      vtxC.y = vtxD.y = tlScaleY( 1.0f );
    } else {
      vtxA.x = vtxC.x = tlScaleX( 0.45f );
      vtxB.x = vtxD.x = tlScaleX( 0.55f );
      vtxA.y = vtxB.y = tlScaleY( 0.45f );
      vtxC.y = vtxD.y = tlScaleY( 0.55f );
    }

    vtxA.tmuvtx[0].sow = vtxC.tmuvtx[0].sow
     = 0.0f;
    vtxB.tmuvtx[0].sow = vtxD.tmuvtx[0].sow
     = 255.0f;
    vtxA.tmuvtx[0].tow = vtxB.tmuvtx[0].tow
     = 0.0f;
    vtxC.tmuvtx[0].tow = vtxD.tmuvtx[0].tow
     = 255.0f;
```

Para establecer el filtro bilineal o *Point Sampled* utilizaremos la función *grTexFilterMode*. Dependiendo del filtro a utilizar se seleccionará un modo diferente:

- Point Sampled: **GR_TEXTUREFILTER_POINT _SAMPLED**
- La Interpolación Bilineal: **GR_TEXTUREFILTER _BILINEAR.**

```
    if ( bilerp ) {
      grTexFilterMode( GR_TMU0,

     GR_TEXTUREFILTER_BILINEAR,

     GR_TEXTUREFILTER_BILINEAR );
      tlConOutput( "GR_TEXTUREFIL-
     TER_BILINEAR    \r" );
    } else {
      grTexFilterMode( GR_TMU0,

     GR_TEXTUREFILTER_POINT
     _SAMPLED,

     GR_TEXTUREFILTER_POINT
     _SAMPLED );
      tlConOutput( "GR_TEXTUREFIL-
     TER_POINT_SAMPLED\r" );
    }
```

Dibujamos los dos triángulos que componen el rectángulo.

```
    grDrawTriangle( &vtxA, &vtxD, &vtxC );
    grDrawTriangle( &vtxA, &vtxB, &vtxD );
    tlConRender();
    grBufferSwap( 1 );
    grSstIdle();
...

  grGlideShutdown();
  return }
```

Las *API's gu3dfGetInfo* y *gu3dfLoad* cargan un fichero *3DF* dentro de la estructura *Gu3DfInfo*. Seguidamente los datos pueden ser extraidos de la misma para inicializar la estructura *GrTexInfo* utilizada en las rutinas de texturas de *Glide*.

```
static int loadTexture( const char *filename,
            GrTexInfo *info,
            GrTexTable_t *tableType,
            void *table ) {
...

  if ( gu3dfGetInfo( filename, &tdfInfo ) ) {
    tdfInfo.data = malloc( tdfInfo.mem
     _required );
    assert( tdfInfo.data );
    if ( gu3dfLoad( filename, &tdfInfo ) ) {
```

Cargamos la información del fichero conteniendo *ratio* de aspecto, nivel de detalle, etc.

```
      info->smallLod   =
     tdfInfo.header.small_lod;
      info->largeLod   =
     tdfInfo.header.large_lod;
      info->aspectRatio =
     tdfInfo.header.aspect_ratio;
      info->format     = tdfInfo.header.
     format;
      info->data       = tdfInfo.data;
      *tableType = texTableType( info->
     format );
      switch( *tableType ) {
       case GR_TEXTABLE_PALETTE:
         memcpy( table, &(tdfInfo.table),
     sizeof( TextureTable ) );
         break;
       default:
         break;
      }
      rv = 1;
    }
  }
  return rv;
}
...
  return rv;
}
```

Con esto hemos finalizado el ejemplo práctico utilizando todo lo aprendido con *Texture Mapping*.

Multimedia con Java: sonido y vídeo (I)

Javier Sanz Alamillo (jsanza@teleline.es)

Con la aparición de las *API's Java Media Framework* y *Java Sound* ha cambiado el panorama multimedia de *Java*. Gracias a ellas ya es posible desarrollar aplicaciones multimedia que utilicen audio y vídeo de una forma sencilla y práctica, sin perder por ello la compatibilidad que siempre ofrece *Java*.

Hasta hace poco las posibilidades multimedia de *Java* eran bastante reducidas. El tratamiento de audio se basaba en poco más que la reproducción de ficheros *AU* de *Sun*, que aunque son muy reducidos de tamaño, no ofrecen más que una salida *mono* y de escasa calidad.

Se abren amplias posibilidades multimedia con las nuevas API´s

Debido a estas carencias, se crearon grupos de desarrollo que generaban librerías para reproducir otros formatos de sonido, como por ejemplo *MIDI*. El hecho de no tener acceso a muchas interioridades de *Java* motivó que los resultados fueran librerías que no eran portables, incluso que no funcionaban en ordenadores de la misma plataforma. En cuanto al vídeo se hacían muchas promesas pero no existía absolutamente nada.

INTRODUCCIÓN

Gracias a *Java Sound* y *Java Media Framework* se abren todas las posibilidades multimedia con *Java*. Mediante *Java Sound* podemos reproducir de una forma sencilla la gran mayoría de los formatos de audio, como *WAV, MIDI, AU*, etc., capturar sonido, reproducir *streams* de audio, etc. Mediante *Java Media Framework*, y más específicamente mediante una de sus partes, *Java Media Player*, podemos utilizar los distintos formatos de vídeo, como *MPEG, AVI, MOV*, capturar *streams* de vídeo y como nó, también se puede reproducir audio, y aunque el tratamiento es algo diferente es igualmente efectivo. Como se aprecia, el conjunto de posibilidades se ha incrementado de manera notable. Gracias a estas dos *API's*, se pueden crear aplicaciones multimedia que hasta ahora se desarrollaban con *Visual Basic, Delphi* o similares, con la ventaja de la portabilidad, la orientación a objetos y la integración con el lenguaje, sin necesidad de utilizar ficheros *OCX* ni similares.

TRATAMIENTO DEL SONIDO EN APLICACIONES MULTIMEDIA

Aquellos que hayan tenido que realizar alguna aplicación que utilizara sonido conocerán las diferentes dificul-

tades que implicaba utilizar la reducida *API* de *Java* disponible. Ya no solamente por los escasos tipos de ficheros de sonido que se podían manejar, por no decir el único ficheros *AU*, sino por la mínima facilidad que ofrecía esta *API* y lo que es casi peor aún, su funcionamiento aleatorio en muchos casos.

Todos hemos llegado a reproducir el tipo de fichero *AU* en un *applet*, pero cuando esto se tenía que hacer en una aplicación algunos habrán tenido que recurrir a "artimañas" como la siguiente:

```
import sun.audio.* ;
import Java.io.*;

class sonido {
 public static void main ( String args[] ) {
  new toca ().start();
    System.out.println (" go on ... ");
 }
}

class toca extends Thread {
 public void run () {
  try {
   while ( true ) {
    AudioPlayer.player.start (new FileInputStream ("d:\\sonido.au") );
    AudioPlayer.player.run();
   }
  }
  catch ( Exception e) {
   e.printStackTrace();
  }
 }
}
```

Para empezar, muchos se habrán asombrado de que se realice un *import sun.audio.**, ya que por una parte, esta aplicación no pasaría el test de *100% Java Pure* y por otra, *sun.** es un paquete que no se debe usar, pues es ahí donde se almacenan las futuras características de *Java* y se van introduciendo poco a poco. En este caso, ni *Sun* garantiza su completo y menos aún su correcto funcionamiento.

Pero para poder reproducir un fichero de sonido en una aplicación sólo se podía realizar mediante el uso de *sun.**, se asumía el riesgo de que no funcionara en otras plataformas y se continuaba.

Todos estos problemas se han terminado con el uso de *Java Sound y Java Media Framework*, olvidándonos de las partes oscuras y del uso de artificios para que una aplicación que utilice sonido funcione correctamente. Las *APIs Java Sound* y *Java Media Framework* permiten un control tanto de reproducción como de captura de audio, reproducción de vídeo, etc., manteniendo todas las características de *Java*. Se pueden ahora reproducir archivos *WAV, AIFF, MIDI*, capturar audio, etc.

Gracias a la API de Java 2 se pueden reproducir una gran variedad de ficheros de audio

Lo más importante de las mismas es que mantienen las características de *Java*. Pero, ¿qué significa exactamente eso? En la actualidad existen muchos sistemas para tratar el sonido en las aplicaciones. Por todos es conocida la inmensa cantidad de ficheros *OCX* para *Visual Basic* que existen en el mercado, pero cada uno de ellos tiene su particularidad, por lo que mientras que en algunos simplemente se le dice qué fichero de sonido hay que reproducir, en otros, se debe afinar un poco más y requiere indicar algunas características del dispositivo. Todo esto sin olvidar que la potencia del ordenador es fundamental para que todo funcione correctamente, ya que en función de un *OCX* u otro, la aplicación se ejecutará con más o menos velocidad. Además, únicamente podemos usar la aplicación en otro ordenador de la misma plataforma.

Con *Java Sound* y *Java Media Player* se eliminan todos estos problemas y se mantiene la compatibilidad entre diferentes plataformas. Además, como el desarrollo utiliza la filosofía *Java*, no tenemos que atender a razones de dispositivos ni similares. Por lo tanto, todos los requerimientos a bajo nivel no son problemas del desarrollador. Como bien sabrá el lector, usando *JDK 1.2* se puede reproducir con cierta facilidad sonido muestreado y sonido *MIDI*, tanto en *applets* como aplicaciones. Por ello, antes de comenzar a describir las *API's Java Sound* y *Java Media Framework*, comentaremos brevemente el uso de *JDK 1.2* para reproducir sonido y así se podrán hacer comparaciones sobre el funcionamiento, modo de desarrollo, facilidades y posibilidades con respecto a estos nuevas *APIs*.

SONIDO MEDIANTE JDK 1.2

Gracias al nuevo *JDK 1.2* o *Java 2*, como el lector prefiera llamarlo, se pueden reproducir diferentes tipos de ficheros de audio, desde los tipos *WAV* hasta los *AU*, tanto en *applets* como en aplicaciones, sin tener que recurrir a una programación complicada. A continuación se van a describir los pasos fundamentales para reproducir sonido mediante *JDK 1.2*, tanto en *applets* como en aplicaciones.

REPRODUCCIÓN DE SONIDO EN UN APPLET

La forma en la que se puede reproducir sonido en un *applet* no ha cambiado en esta versión de *JDK*, aunque ahora se pueden reproducir, por ejemplo, ficheros de sonido *WAV*. Para reproducir un fichero de audio, simplemente se invoca:

```
Applet.getAudioClip();
```

y mediante el objeto *AudioClip* generado se puede llamar a métodos

como *start(), stop(), loop()*. El fichero de audio es cargado cuando se contruye *AudioClip*. La carga no es asíncrona, así que si necesita que lo sea, simplemente deberá incluir la reproducción en un hilo diferente. Con independencia del tipo de fichero de audio que reproducir, el código que lo ejecuta es el mismo, lo cual ofrece una independencia importante a la aplicación sobre posibles modificaciones que se pudieran realizar.

REPRODUCCIÓN DE AUDIO EN UNA APLICACIÓN

J*DK 1.2* ofrece un nuevo método estático incluido en el *paquete Java.applet.Applet* denominado *newAudioClip* que permite a las aplicaciones reproducir audio mediante la indicación de una *URL*. Más específicamente, el método tiene la siguiente sintaxis:

```
public static final AudioClip newAudioClip(URL r)
```

Para reproducir un fichero de sonido, se siguen los pasos anteriormente descritos:

1) Se invoca *Applet.newAudioClip()* y se le pasa un objeto *URL* con la dirección adecuada.
2) Se llama al método *play()* o *loop()* de *AudioClip*.

Si bien mediante estos métodos se pueden reproducir ficheros de audio, encontraremos que hay ciertas dificultades para reproducir algunos ficheros *WAV* y *AU* que hayan sido comprimidos mediante *ADPCM* o cualquier otro tipo de compresión. Además, la calidad de los ficheros *AU* es algo reducida, pudiéndose escuchar ciertos ruidos.

Ahora bien, que contestación se obtiene de las siguientes preguntas. ¿Cómo puedo aumentar el volumen de reproducción usando la *API* de *Java 2*? ¿y si sólo quiero que suene un determinado altavoz? Mediante la *API* actual de *Java 2* no se pueden realizar tareas similares a ésta, pero gracias a estos nuevas *API*, implementar esos requerimientos es una tarea rápida, sencilla y eficaz.

Debido a que *Java Sound* pronto formará parte de *JDK 1.2* en una de sus próximas *releases*, y que se liberará una versión definitiva de la propia *API* (actualmente es la versión 0.8) sin por ejemplo, los mensajes de *debug* actuales y algunas deficiencias más o menos determinadas, podremos disponer de todas esas nuevas posibilidades utilizando la *API Java 2*.

Ya no hace falta recurrir a determinados artificios para reproducir sonido

De todos modos, no hace falta esperar a su lanzamiento, puesto que utilizando la versión actual de *Java Sound* y uniendo a esto todas las posibilidades que ofrece *Java Media Player*, se disponen de los medios necesarios (y más que suficientes) para crear aplicaciones multimedia usando *Java*. A lo largo de esta serie de artículos se van a describir detalladamente todas las posibilidades que ofrecen *Java Sound* para reproducir sonido, ya sea muestreado o *MIDI*, y *Java Media Framework*, más específicamente en una de sus partes, *Java Media Player*, con el que podemos reproducir fácilmente vídeo en distintos formatos, además de poder utilizar las mismas posibilidades en audio.

JAVA SOUND API

J*ava Sound* es una *API* que ofrece la posibilidad de reproducir audio muestreado y *MIDI* con una cierta calidad, gracias a un nuevo *engine* de audio de 32 *bits*. Permite reproducir una gran variedad de formatos de audio, como se ha comentado, desde *AIFF, AU, WAV* además de *MIDI* de los tipos *TYPE0, TYPE 1* y *RMF*. Además, permite tratar audio de 8 o 16 *bits*, en mono o estéreo, muestreado entre las frecuencias de *8kHz* y 48Khz y audio en forma de *stream*.

Hay que recordar que *Java Sound* inicialmente es distribuido como una *API* independiente de *JDK*. Mientras que en la versión 1.2 se ha integrado ya la interfaz *AudioClip*, con la que se pueden reproducir la mayoría de los formatos de audio, la totalidad de la *API* de *Java Sound* se integrará en una próxima *release* de *JDK 1.2*. A continuación se van a tratar las características principales de la arquitectura de *Java Sound*:

ARQUITECTURA DE LOS DISPOSITIVOS

J*ava Sound* define dos tipos de dispositivos:

- Dispositivos del sistema.
- Dispositivos de control.

Se puede definir el dispositivo del sistema como aquel que se encarga de todo tipo de gestión con una parte del *hardware* o de un recurso del sistema que se vaya a utilizar. Un dispositivo de control ofrece toda una serie de interfaces para realizar una tarea sobre los datos de audio, ya sea ésta de reproducción o captura, e incluso en función del recurso, se puede controlar directamente el recurso *hardware*. Además, se puede disponer de muchos dispositivos de control para un mismo dispositivo del sistema. Por ejemplo, un dispositivo de control que puede representar a la vez un sintetizador *MIDI*, un mezclador o un reproductor de *CD* a través del mismo dispositivo del sistema.

Un *SystemDevice* o dispositivo del sistema puede usar uno o más *Ports* o puertos de entrada/salida, que representan la fuente de datos o el destino final de los mismos. Por ejemplo, un dispositivo de sistema *MIDI* consta de dos puertos, *MIDIIn* y *MIDIOut* y también como puertos se suelen identificar

¡NO TE LO PIERDAS!

2495 ptas.
IVA incluido

DIRECTORIO GENERAL
INTERNET '99

PREVISUALIZACIÓN DE LAS PÁGINAS Y CONEXIÓN DIRECTA DESDE EL CD-ROM

* INCLUYE BUSCADOR PARA EL CD-ROM
* ESPECIAL ÉNFASIS EN WEBS EN ESPAÑOL
* AHORRE HORAS DE BÚSQUEDA EN INTERNET

CONTIENE CD-ROM

Ahorra horas de busqueda en Internet

Especial énfasis en Webs en español



C/ Aragoneses, 7 – 28108 Pol. Ind. Alcobendas (MADRID)
Tel.: 91 661 42 11* – Fax: 91 661 43 86
http://www.abetoed.es

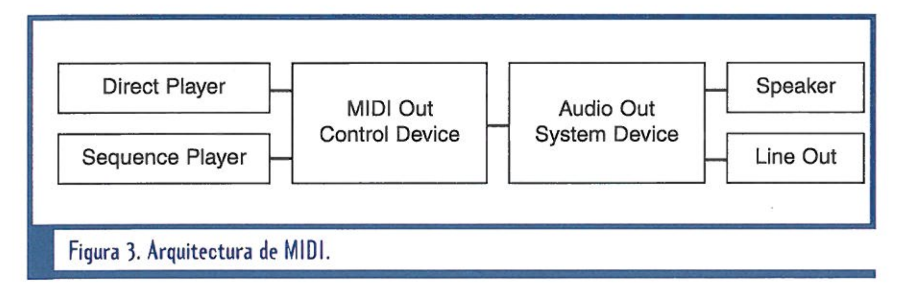


Figura 3. Arquitectura de MIDI.

controla y gestiona la salida del recurso tipo *MIDI*. Ésta puede o no estar gestionada mediante el mismo *SystemDevice* usado para el muestreo de la salida de audio. Se puede consultar el estado del *MIDIOutControlDevice* para averiguar el número de voces disponibles. Además, permite la generación de dos tipos de objetos *Channels*, que soportan el interface *MIDIPlayer*. Un objeto tipo *MIDIPlayer* es en sí el software de *MIDI*. Realiza todas las tareas relativas a las voces y modos *MIDI*.

Gracias a la clase audiomanager se puede acceder a las posibilidades de Java Sound

Un objeto tipo *SequencePlayer* es un *MIDIPlayer* con posibilidades de secuenciado. Permite gestionar *Buffers* que contienen *MIDIMessages* mediante el uso de la interfaz *PushSink*. Es mediante este tipo de objeto y el método *loadSequence* como se realiza la carga de un fichero *MIDI*, aunque *SequencePlayer* ofrece métodos para iniciar y detener la reproducción de un *MIDI*.

En la figura 3 se muestra la arquitectura de uso del formato *MIDI*.

Hasta ahora se han presentado los conceptos básicos que tener en cuenta para desarrollar aplicaciones multimedía con *Java Sound*. En el próximo artículo se desarrollarán varios ejemplos sobre como reproducir audio muestreado y *MIDI*, se describirá la captura de audio y se introducirán el conjunto de nuevas posibilidades de *Java Media Framework*, tanto para manejar vídeo como audio.

CAPTURA DE AUDIO

También mediante *Java Sound* se puede capturar audio. En la figura 4 se muestra la arquitectura relativa a este aspecto.

Mediante un objeto *AudioInControlDevice,* del tipo *ControlDevice,* se puede capturar audio. Se realiza mediante los canales de entrada pudiéndose usar los modelos *pull* y *push.* Cada canal de entrada consta de un objeto *Format* y se puede comprobar el estado de la captura, eventos de inicio y final, así como el control de volumen mediante *GainControl*.Un objeto tipo *PushAudioInput* se conecta a otro *PushSink* y le añade datos. La operación *write* es crítica al igual que resultaba con *read* en la reproducción. Nadie espera que en la captura de audio haya pausas extrañas. Por ello, toda operación *write* es ejecutada inmediatamente, por lo que la aplicación no debe bloquearse o realizar otros procesos durante esta operación. Cuando no hay más datos que capturar, *PushSink* finaliza, indicando el número de *bytes* procesados y devolviendo ese valor. Las aplicaciones que utilizan el modelo *pull* deben realizan todo el procesamiento o bloquearse durante la ejecución de *write.* En este caso, *PullSink* se conecta a *PullAudioInput* y recoge los datos del *buffer* donde se almacenan.

Como se puede observar, la filosofía de trabajo en la captura de datos es similar a la utilizada en la tarea de reproducir audio, por lo que su aprendizaje no requiere mucho más esfuerzo.

Hasta ahora se han presentado los conceptos básicos necesarios que se deben tener en cuenta para desarrollar aplicaciones multimedia con *Java Sound*. En el próximo artículo se desarrollarán varios ejemplos sobre cómo reproducir audio muestreado y *MIDI*, y se introducirán el conjunto de nuevas posibilidades de *Java Media Framework* (en la parte *Java Media Player)* tanto para manejar vídeo como audio.

CONCLUSIONES

Gracias a las nuevas API's *Java Sound* y *Java Media Framework* (*Java Media Player*), se pueden desarrollar aplicaciones que utilicen audio y vídeo de una forma práctica y eficaz, manteniendo todas las características propias de *Java*.

Mediante *Java Sound* podemos manejar audio, tanto para reproducir como para capturar, mediante el uso de unos pocos objetos, pudiendo acceder así a nuevas posibilidades que no se definen en la *API* de *Java* 2.

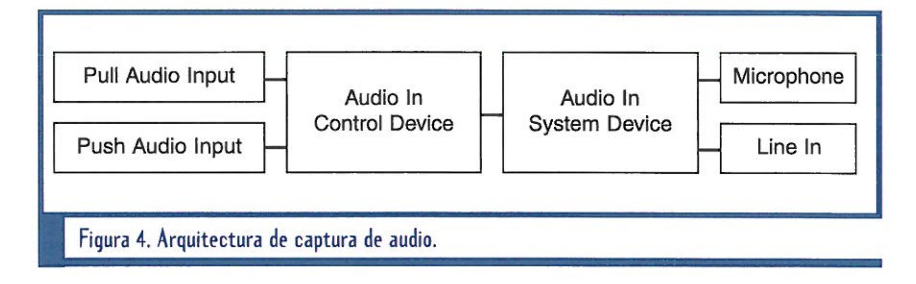
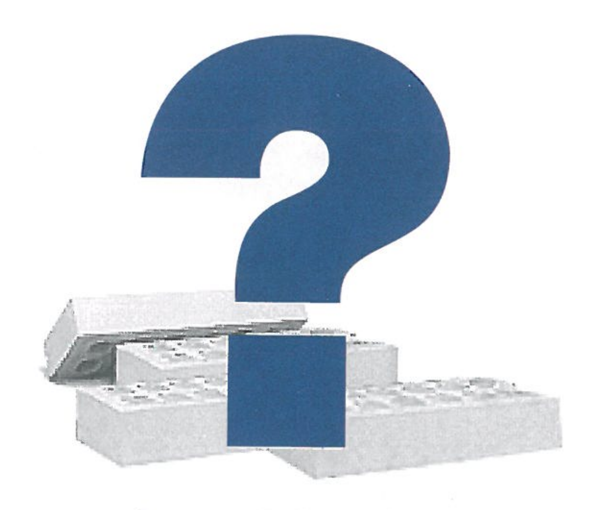


Figura 4. Arquitectura de captura de audio.

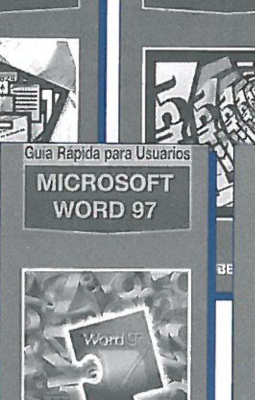


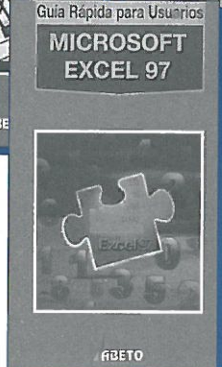
¿ Realmente necesitas un manual de más de 500 páginas para sacar mayor partido a tu ordenador?

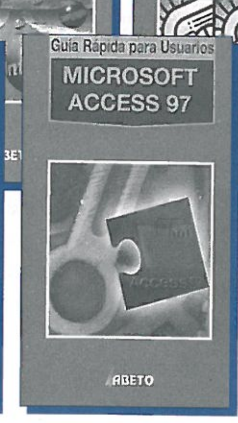
La colección **Guías Rápidas para Usuarios** te ofrece soluciones **PRÁCTICAS**













Guía Rápida para Usuarios — NAVEGAR EN INTERNET

Guía Rápida para Usuarios — POWERPOINT 7.0 PARA WINDOWS 95

Guía Rápida para Usuarios — COREL WORDPERFECT 7.0 PARA WINDOWS 95

Cada libro por sólo

995 ptas. iva incluido

LO IMPORTANTE ES LO ESENCIAL

Distribuidores autorizados

COMUNIDAD DE MADRID / CASTILLA LA MANCHA
DISTRIFORMA S.A. — Tfno. 91 - 501 4749
CATALUÑA
MIDAC LLIBRES S.L. — Tf. 93-421 18 95
ANDALUCIA OCCIDENTAL/ EXTREMADURA
DISTRIBUCIÓN DE EDICIONES RDGUEZ. SANTOS S.L. — Tfno. 95 - 418 04 75
ANDALUCIA ORIENTAL
DISTRIBUCIONES DEL MEDIODIA S.A. (ZÓCALO) — Tfno. 958-550278
CASTILLA - LEÓN
ARCADIA S.L. — Tfno. 983-395049
GALICIA
LUIS REY ABELLA (DISGALIBRO) — Tfno. 981-795754
ASTURIAS - CANTABRIA
DISTRIBUCIONES CIMADEVILLA S.A. — 98-5167930

CANARIAS
GARCIA PRIETO LIBROS S.L. — Tfno. 922-820026
BALEARES
PONENT LLIBRES S.L. — 971-430339
VALENCIA - CASTELLÓN
ADONAY S.L. — 96-3975148
ALICANTE - MURCIA - ALBACETE
LA TIERRA LIBROS S.L. — Tfno. 96-5110192
PAÍS VASCO - NAVARRA
YOAR S.L. — Tfno. 948-302239
ARAGÓN - LA RIOJA
ICARO DISTRIBUIDORA S.L. — Tfno. 976-126333



Tel.: (91) 661 42 11*

Del HTML al acceso a bases de datos (y II)

Juan Manuel Menéndez (jfrias@ibm.net)

Para concluir con esta serie de artículos vamos a tratar de aprender cómo se crea un sistema de acceso en tres capas. También haremos una introducción acerca de unos interesantes elementos: hablamos de los servidores de aplicaciones.

DE LA TEORÍA A LA PRÁCTICA

En el anterior capítulo vimos cómo funciona un aplicación *CGI*, tanto a nivel teórico como práctico mediante un pequeño ejemplo. La parte final explicaba las características de una aplicación *intranet* de tres capas.

El inicio de un programa en un sistema implica que éste debe crear un proceso cuya realización suele ser una tarea pesada

En este segundo y último artículo de la serie realizaremos un sistema de acceso a datos en tres capas, para después poder analizar los contras que tiene la realización práctica de este sistema y acabar introduciendo un nuevo elemento (los servidores de aplicaciones) que aparece en las aplicaciones *intranet.*

REALIZACIÓN DE UN MODELO DE TRES CAPAS

En el modelo que se va a realizar el navegador descargará una página del servidor *Web* que contendrá un formulario en el que se deberán introducir ciertos datos: nombre, *e-mail* y comentarios. Esta información se enviará al servidor que los guardará en una base de datos. Para llevar a cabo este sistema se precisan los elementos siguientes:

- Un programa en *C++* (**cgi1.exe**)
- Un programa en *Visual Basic* (**cgi2.exe**)
- Una base de datos *Access* (**cgi.mdb**)
- Un documento *HTML* (**formulario.html**)

El programa **cgi1.exe** fue explicado en parte en el número anterior, sobre todo en lo relativo al funcionamiento de *CGI*, aunque no se comentó el proceso que se realiza con los datos procedentes del formulario. Estos datos son escritos en un fichero denominado **CGIx.HFO**, donde **x** se refiere a un número entero que comienza por **0**, siendo el programa el encargado de generarlo. En la primera ejecución el fichero tendrá el nombre **CGI0.HFO**, pero si ya existe otro fichero, normalmente producto de otra ejecución anterior mal finalizada, el sistema utilizará el nombre **CGI1.HFO** y así sucesivamente. En las líneas siguientes se muestra el código encargado de efectuar este proceso.

```
for (i = 0; i <= 9999; i++)
  {
    // Make a new filename
```

```
    sprintf(FileName, "CGI%d.HFO", i);

    // Si el fichero exite sale del bucle
    if(access(FileName, 0) == -1)
        break;
}
```

Una vez creado el fichero se graba en él cada uno de los parámetros introducidos en el formulario por el cliente:

```
URL=14.0.0.1
Date=10/02/99
Time=22:45:00
Name=Juan Manuel Menéndez
E-mail=jfrias@ibm.net
Comments=Esto es un comentario
```

Este fichero es leído por el programa **cgi2.exe** y borrado posteriormente, aunque si se produce un error durante la ejecución estos ficheros no se borran.

Un servidor con muchos procesos está más ocupado en gestionarse a si mismo que en atender a las peticiones de los clientes

El programa **cgi2.exe** está codificado en *Visual Basic*, pero de una forma un tanto especial pues no abre ningún formulario. La lógica de este programa es bastante simple:

- Lee todas las líneas del fichero *HFO*.
- Separa el valor del parámetro del nombre del parámetro utilizando el carácter "=".
- Instancia un objeto base de datos y un objeto *recordset* asociado a una tabla de la citada base de datos.
- Inserta en la tabla un nuevo registro mediante el método *addnew*.

```
Private Function ParseField(szText As String)
    As String
Dim k As Integer
```

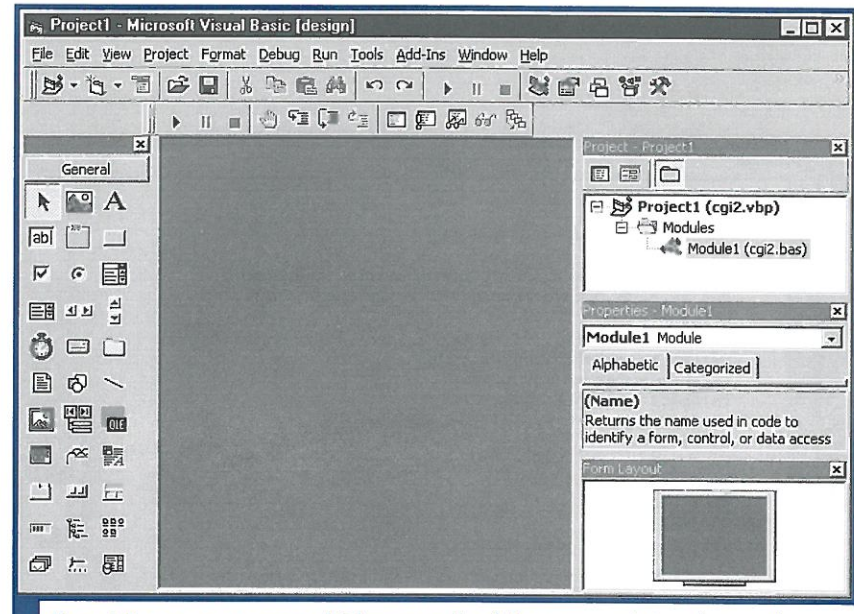


Figura 1. Este es un caso especial del programa Visual Basic, carece de formularios, tal y como se aprecia en las ventanas.

```
k = InStr(szText, "=")
ParseField = Mid$(szText, k + 1)
End Function
```

Apertura de la base de datos:

```
Set db = OpenDatabase(App.Path &
    "\cgi.mdb")
Set rs = db.OpenRecordset("table1",
    dbOpenTable)
```

Inserción del nuevo registro y cierre de la base de datos:

```
rs.AddNew
rs!When = ParseField(szDate) & " " &
    ParseField(szTime)
rs!URL = ParseField(szURL)
rs!Name = ParseField(szName)
rs!Email = ParseField(szEmail)
rs!Comments = ParseField(szComments)

' Actualizar la tabla, cerrar todo y salir
rs.Update
rs.Close
db.Close
```

Como el lector puede observar la misión del fichero **CGIx.HFO** es la de servir de medio de transferencia de información entre el programa **cgi1.exe** y el programa **cgi2.exe**.

¿QUÉ HA HECHO EL SISTEMA?

Vamos a ver qué tareas ha realizado el sistema operativo. El servidor *Web* una vez que ha detectado que la petición requiere la ejecución de un programa arranca el programa solicitado. El hecho de arrancar un programa en un sistema implica que el sistema debe de iniciar un proceso donde se va a ejecutar dicho programa y este trabajo suele ser de las tareas más "pesadas" para un sistema operativo El inicio de un nuevo proceso supone los siguientes pasos:

- Creación del nuevo espacio de memoria para ese proceso.
- Asignación de una nueva tabla de direcciones.
- Modificación de gran número de variables internas del *kernel* para que contemplen la existencia del nuevo "inquilino" en la máquina.

Gran parte de estas tareas bloquean el resto los procesos, pues mientras se modifican ciertos valores internos del *kernel* el resto de los procesos quedan a

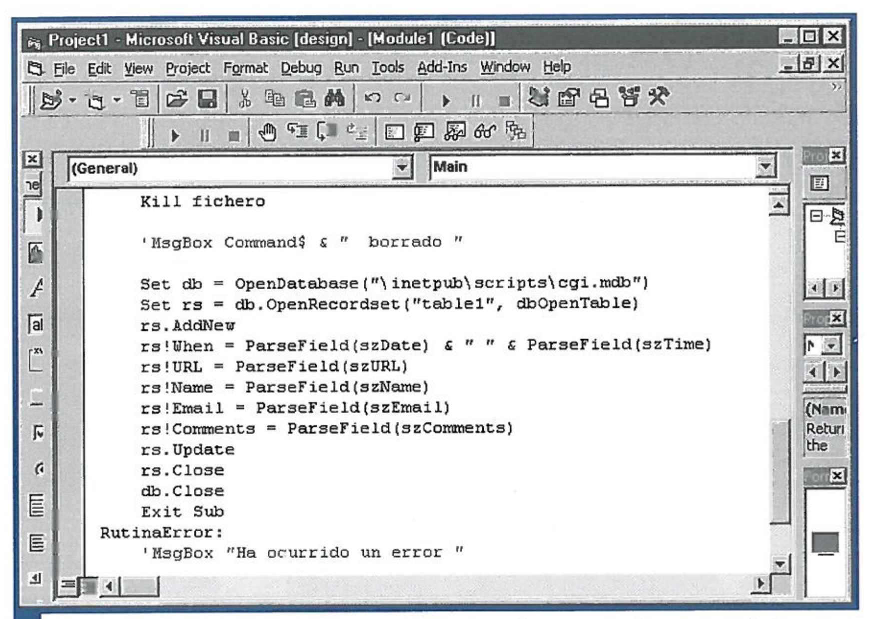


Figura 2. Este módulo de Visual Básic está dedicado exclusivamente a la gestión de la base de datos, la parte de tratamiento con Internet es realizada por el módulo cgi1.exe.

la espera de poder ejecutarse. Por otra parte, la existencia de un elevado número de procesos en los sistemas trae consigo una degradación del mismo cuyo resultado es una función logarítmica, y además tiene un consumo directo de memoria real y de fichero de *swap*. A esto hay que sumar las tareas que debe realizar el proceso cuando finaliza.

Si existe un servidor con muchos procesos en el sistema y alta volatilidad de los mismos, es decir un servidor con un promedio de procesos arrancados alto y al mismo tiempo un gran número de los mismos creándose y destruyéndose, tenemos la típica situación de que esté más ocupado en gestionarse a si mismo que en atender a las peticiones de los clientes. Y esta situación es a la que se llega cuando se tiene un servidor *Web* basado en *CGI* con un elevado número de transacciones en ejecución. En el sistema que se ha estado desarrollando a efectos prácticos se crean dos procesos por cada petición basada en *CGI* que llega desde un navegador, pero si se ejecuta todo en el mismo programa se rompe la arquitectura cliente/servidor de tres capas y con esta rotura se pierde ese conjunto de ventajas que se han descrito previamente. El siguiente problema está relacionado con el acceso a las bases de datos, en concreto que la apertura del mismo y una vez más en este proceso se vuelve a dar a otra escala la misma lista de tareas:

- Apertura de una base de datos.
- Ejecución de instrucciones contra la misma.
- Cierre de la base de datos.

De esta lista de tareas sólo dos existen porque no hay una conexión previa con la base de datos y dadas las características del *CGI* la otra tarea suele ser corta, generalmente limitada a una instrucción *SQL* o lo más a la ejecución de un procedimiento catalogado en la misma. Por último y para finalizar indicaremos que en este caso y por simplificar se ha elegido un método fácil (pero poco elegante desde el punto de vista de un programador) de comunicación entre los dos programas. Además de esta pequeña desventaja, de nuevo se vuelve a realizar el mismo proceso, sólo que esta vez en el acceso al disco:

- Creación de un fichero.
- Escritura de la información por el primer programa.
- Lectura de la información por el segundo programa.
- Borrado del fichero.

Es seguro que el sistema ha empleado más acceso a disco para realizar la creación y borrado del fichero que para la lectura y escritura de la información. Bueno pues en este momento ya está el drama escrito y sólo queda ver el desenlace, siendo la solución más inmediata la que consiste en eliminar el modelo de tres capas e integrar el acceso a datos en el mismo programa *CGI*. No se soluciona el problema de la conexión a la base de datos pero sí el relativo a la creación de un proceso, así como la necesidad del sistema de comunicación entre ambos programas.

Mientras se ejecutan valores internos del Kernel el resto de procesos esperan para ejecutarse

En este caso y para no renunciar a la simplicidad del planteamiento inicial, se intentará afinar un poco más para encontrar otra solución intermedia. La solución pasa por evitar que el programa de acceso a datos tenga que ser arrancado y parado cada vez que recibe una petición de acceso a datos. La primera ventaja es inmediata, ya que se reduce a la mitad el número de procesos arrancados, aunque aparece un nuevo problema que es el de la transferencia de información entre el programa **cgi1.exe** y **cgi2.exe**. Es posible eliminarlo estableciendo un sistema de comunicación interno entre programas como puede ser un *socket*. Si se decide optar por una comunicación de este tipo, nos encontraremos con una pequeña anomalía con respecto a otros sistemas cliente/servidor en los que normalmente el servidor está escrito en *C* y el cliente en *Visual Basic*. La parte servidora quedaría así:

```
Public Sub main()
Dim....
```

```
Winsock1.LocalPort = 10000
Winsock1.Protocol = sckTCPProtocol
Winsock1.Listen
```

La parte del cliente quedaría así:

```
struct sockaddr_in     ser_addr,cli_addr
if((nsfd = connect(sfd,(struct sockaddr
        *)&ser_addr,sizeof(ser_addr)))==-1)
{
        perror("Error al dar connect:");
        exit (-1); }
printf("\nConexion abierta");
gethostname(nodo_addr,sizeof(nodo_addr));
i = write(sfd,nodo_addr,strlen(nodo_addr));
if (i < 0)
{
        printf("i: %d\n",i);
        perror("Error en send: "); }
printf("\nSend completo: %d caracteres
        enviados",i);
```

Dos de los tres elementos que afectaban al rendimiento de la aplicación *CGI* han sido eliminados. El tercero tiene una fácil solución que consiste en abrir una conexión al inicio del programa con la base de datos y ejecutar todas las peticiones a través de la misma.

EL PROTOCOLO NO ORIENTADO A CONEXIÓN

Partiendo de un diseño inicial hemos depurado el sistema sin renunciar al objetivo de la aplicación ni al modelo de tres capas hemos conseguido otro diseño mucho más efectivo de cara al uso de recursos. ¿Hemos resuelto todos los problemas? No, ya que aún queda un último problema relacionado con una parte del diseño del protocolo *HTTP* en el que se apoya *CGI*. Ilustraremos con un ejemplo el problema. Supongamos que estamos en un sistema de reservas de viaje, en dicho sistema se puede, en una misma transacción, reservar un billete de avión, una habitación de hotel y alquilar un coche. El cliente accede al servicio y después de ver las diversas opciones selecciona un vuelo mediante un formulario *CGI*. A continuación accede a la consulta de hoteles y realiza una reserva en un hotel determinado y ésta es rechazada porque no hay plazas disponibles. Nuestro personaje lo intenta en otros con idéntico resultado y decide no alquilar la habitación. En este punto lo lógico es que el sistema, al ver que el cliente abandona sin reservar habitación, le ofrezca la oportunidad de anular la reserva de avión. Ahora se necesita:

- Tres de los formularios *HTML* (**vuelos.htm, hoteles.htm, coches.htm**).
- Una base de datos con tres tablas: **vuelos, hoteles y coches.**
- Los dos programas: **cgi1.exe** y **cgi2.exe.**

Cada formulario enviará la información necesaria para realizar una consulta. Cuando el usuario se conecta al sistema de reservas su navegador descarga el formulario *HTML* y se cierra la conexión con el servidor *Web*, pues el protocolo *HTTP*, es un protocolo no orientado a la conexión y tras descargarse el documento finaliza la sesión. Esto no quiere decir que se haya cortado la comunicación con nuestro proveedor de *Internet*, sino que cada petición efectuada por el navegador requiere efectuar una conexión mediante el protocolo *HTTP*.

Los servidores de aplicación gestionan toda la problemática que existe en la Red

Hasta aquí no hay problema pues los datos necesarios se recibirán cuando se envíe la siguiente petición al servidor. El usuario elige un vuelo y mediante un botón del formulario envía esos datos al servidor. Entonces el sistema crea una nueva conexión por la que se envían los datos y se arranca el programa **cgi1.exe.** Éste pasa la información al programa **cgi2.exe** y el resultado es devuelto al programa **cgi1.exe**, que prepara la salida en formato *HTML* para el cliente y finaliza su ejecución. El cliente quiere seleccionar ahora un hotel y al igual que ha ocurrido con la reserva de vuelo se ejecutan los programas **cgi1.exe** y **cgi2.exe**, y además varias veces, durante la búsqueda de alojamiento que está realizando el cliente.

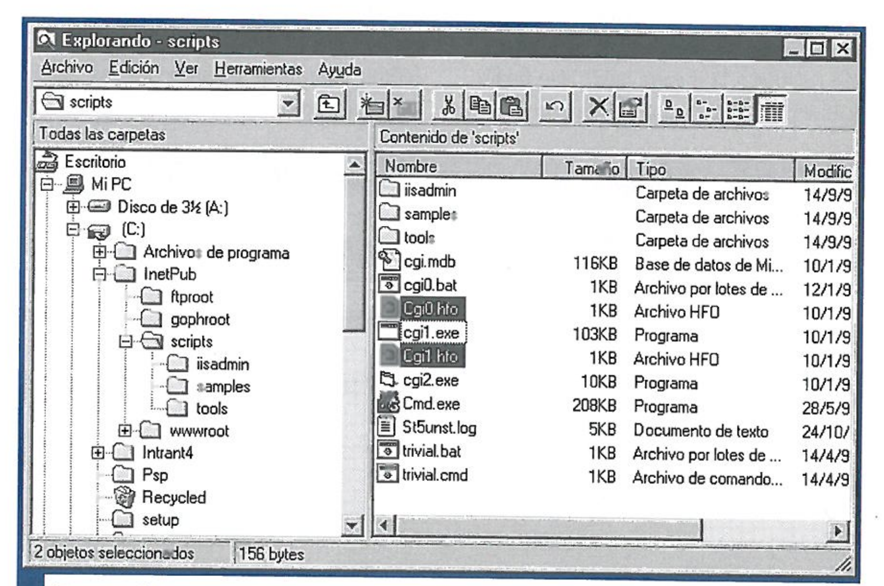


Figura 3. El programa cgi1.exe. va generando ficheros con el nombre cgx.hfo, es una forma simple de conseguir nombres de ficheros distintos.

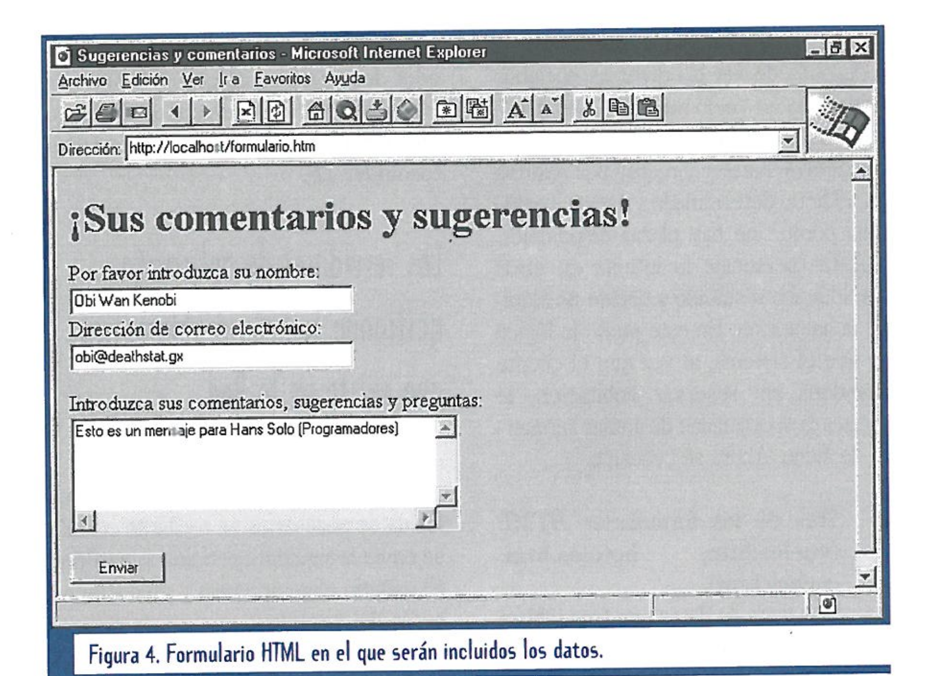


Figura 4. Formulario HTML en el que serán incluidos los datos.

Bueno es el momento de tener la cortesía de ofrecerle al cliente la posibilidad de anular la reserva de vuelo. En este punto se presenta un pequeño problema ¿cuál ha sido la reserva de vuelo que ha hecho el cliente?, ¿cómo podemos, entre todos los registros que hay en la tabla de reservas, localizar la que ha hecho este cliente en particular?

De una forma u otra hay que hacer algo en la aplicación para que la transacción con el cliente permanezca activa, es decir que no se pierda toda la información relativa a la misma hasta que finalice, independientemente de las veces que el protocolo *HTTP*, empiece y acabe una conexión. En el sistema que estamos realizando estos datos de la transacción tienen que ser guardados necesariamente por el programa **cgi2.exe**, porque el otro programa empieza y acaba con cada petición que realiza el formulario. Una vez que tenemos solucionado el problema de mantener los datos durante la vida de la transacción, se debe pensar que el sistema puede estar atendiendo a más de un cliente, es decir, estarán guardados los datos de varias transacciones que se están ejecutando simultáneamente, ¿cómo identificar plenamente los datos de cada transacción teniendo en cuenta que en cada petición el *HTTP* realiza varias conexiones independientes?

LA TRANSACCIÓN COMO UNIDAD DE MEDIDA

Una primera aproximación a la resolución del problema viene al observar que a pesar de todo hay algo que se mantiene invariable a lo largo de toda la transacción que es la dirección *IP* del cliente, y además como se explicó en el artículo anterior este valor siempre es suministrado por el protocolo *CGI* mediante variables de entorno.

Hay un estándar de seguridad soportado por servidores Web y por navegadores basados en HTTP

En fin, parece que el problema está solucionado, por lo que el programa **cgi2.exe** deberá de realizar además lo siguiente:

- Almacenar los datos de la transacción en una tabla temporal, cuya clave es la dirección *IP* del puesto cliente.
- Si la transacción es aceptada los datos se actualizarán en la tabla definitiva y si es rechazada se borrarán de la tabla temporal.

Se complica un poco el programa pero se soluciona el problema, o al menos eso creemos. Supongamos que el cliente se conecta al sistema de reservas desde un equipo conectado mediante módem a *Internet*, pero ¿y si el cliente realiza la reserva desde su puesto de trabajo, o desde una central de reservas, cuyos puestos están en una red de área local conectada a

Datos enviados - Microsoft Internet Explorer
Archivo Edición Ver Ir a Favoritos Ayuda
Dirección: http://venezia/Scripts/cgi1.exe
Su información ha sido aceptada.
¡Gracias!
Return

Figura 5. Formulario de retorno del proceso de datos.

Internet por medio de un *proxy*? Para este caso todos lo puestos de una red de área local que salen a *Internet* por medio de un *proxy*, de cara a la red tienen la misma dirección *IP*. Bueno esto ya invalida el sistema que hemos planteado previamente, puesto que se basaba en la exclusividad de la dirección *IP*.

El modelo de tres capas queda desfigurado al sobrecargar el módulo cgi2.exe

Bueno quedan otras herramientas, ya que se puede enviar al puesto cliente un *cookie*. Con cada *cookie* se puede tener identificado plenamente el puesto cliente, siempre y cuando seamos capaces de generar uno distinto para cada transacción. La aplicación debe realizar ahora los pasos siguientes:

- Pedir el *cookie* si no se está al inicio de la transacción, si por el contrario se está al inicio generarlo.
- Enviar el *cookie* al cliente.
- Almacenar los datos de la transacción en una tabla temporal, cuya clave es la dirección *IP* del puesto cliente.
- Si la transacción es aceptada los datos se actualizarán en la tabla definitiva y si es rechazada se borraran de la tabla temporal.

El programa como se puede ver se va complicando y complicando, y todavía no se ha llegado al final, porque muchos navegadores por temas de confidencialidad están configurados para no aceptar *cookies*. Si después de todo este esfuerzo de programación resulta que el navegador rechaza el *cookie*...

Queda un último recurso y es el de usar certificados de autorización, mediante esta técnica cliente y servidor intercambian unos certificados mediante los cuales se identifican plenamente. No se trata de *cookies* que puedan poner en peligro la intimidad del cliente sino de un estándar de seguridad que soportan tanto los servidores *Web* como los navegadores basados en *HTTP*. Quizá se pregunte el lector: si ya existe una solución para el problema de las transacciones interactivas, basada además en un estándar ¿por qué se ha andado con parches, primero basándonos en direcciones y posteriormente en *cookies*? La respuesta es simple y en este caso sólo se persiguen fines didácticos, con el fin de observar el largo camino recorrido por los sistemas interactivos basados en servidores *Web*.

UN NUEVO INVITADO

Ya parece conseguido el objetivo, pero se ha pagado un precio por ello, un coste que ha corrido a cargo del modelo de tres capas. Un modelo que ha quedado algo desfigurado, pues se ha sobrecargado el modulo **cgi2.exe** es decir el módulo de acceso a datos con tareas que poco tienen que ver con su misión original, pero al ser el único módulo que está activo durante la ejecución de la transacción no existía otra alternativa. Otra alternativa es crear un módulo intermedio entre **cgi1.exe** y **cgi2.exe** algo así como un módulo **cgi1-2.exe**, este modulo se encargaría de las siguientes peticiones:

- Generar los certificados de autorización.
- Ordenar la escritura de los datos en la tabla temporal.
- Mantener el estado de todos los clientes conectados.
- Determinar cuando una transacción ha finalizado.
- Dar orden para el borrado o escritura definitiva de la tabla temporal según sea el resultado de la transacción.
- Llamar a los módulos de aplicación necesarios.

En otras palabras crear un programa que gestione toda la problemática que existe en la red. A los programas de este tipo se les denomina servidores de aplicación. Existen en el mercado varios servidores de aplicación y prácticamente todos los fabricantes de *software* con intereses en la Red están aportando sus soluciones de servidores de aplicación. Cómo puede ser *Netscape* y su producto *Netscape Application Server (NAS)*, que junto con su servidor *Web Netscape Enterprise Server, (NES)*, tiene un solución completa en dos de las tres capas.

Por otra parte están alguno de los fabricantes de sistemas de bases de datos que como *Oracle* han creado un modelo completo:

- *Oracle8i* Como sistema de bases de datos.
- *Oracle Application Server (OAS)* como servidor de aplicaciones.
- *Oracle Web Server (OWS)* como servidor *Web*.

Existiendo en este caso, como es de esperar, una interacción total entre todos los componentes de la familia.

CONCLUSIÓN

Con este segundo capítulo finaliza esta serie de artículos que partiendo de las nuevas necesidades que están surgiendo en la red ha pretendido mostrarlo siguiente:

- Interactividad con el usuario.
- Acceso a datos.
- Sistemas transaccionales.

La dificultad que entraña la puesta en la red de los nuevos servicios y así poder comprender un poco mejor las nuevas tecnologías que vienen, algunas de éstas serán sobre las que se tratará más en profundidad en el futuro, seguro.

Desarrollo de aplicaciones con videoconferencia (II)

Constantino Sánchez Ballesteros (constantino@nexo.es)

Como prometimos en el artículo anterior, abordaremos un proyecto simple de conferencia mediante el uso de *scripts* de *Visual Basic* aplicados en páginas *web*.

MULTIPOINT DATA CONFERENCING

Una de las características más notables de *NetMeeting* es el soporte multipunto en las conferencias, el cual permite comunicarnos y colaborar con otras personas en tiempo real sobre *Internet* o redes de área local *NetMeeting* permite compartir aplicaciones, intercambiar información entre aplicaciones compartidas a través del portapapeles, transferir archivos, colaborar con una pizarra y comunicarse a través de *chat*. Además, el soporte del estándar *T.120* de conferencia de datos permite a *NetMeeting* trabajar con otros productos basados en este estándar. Las siguientes características detallan la tecnología multipunto:

- **Compartir aplicaciones.** Podemos compartir con otros participantes un programa en ejecución sobre un ordenador. Los participantes pueden visualizar los mismos datos o información, y ver las acciones al igual que lo haría la persona que comparte la aplicación.
- **Compartir portapapeles.** El portapapeles compartido ofrece la posibilidad de intercambiar contenidos con otras personas en la llamada utilizando los comandos cortar, copiar y pegar. Podemos copiar información de un documento local y pegar su contenido dentro de una aplicación compartida. Esta capacidad permite el mismo intercambio de información entre aplicaciones compartidas y aplicaciones locales.

Para usar un control VBScript hay que referenciar el identificador

- **Transferencia de archivos.** Mediante la transferencia de archivos se puede enviar un fichero a una o a todas las personas unidas en la conferencia. Mientras se efectúa la transferencia de archivos, podemos seguir utilizando las demás características de *NetMeeting* como *chat*, pizarra, etc.
- **Pizarra.** Este programa es una aplicación multipágina y multiusuario que permite dibujar, escribir o visualizar cualquier información gráfica. La pizarra está orientada al objeto, permitiendo movernos y manipular el contenido.
- **Chat.** Podemos escribir mensajes de texto para compartir ideas comunes con los demás participantes. Podemos utilizar el *Chat* cuando no hay de sonido o vídeo.

UTILIZANDO NETMEETING EN PÁGINAS WEB

Con el uso de *scripts* para páginas *web* creamo programas no compilados que funcionan como si se hubiesen creado con la ayuda de un compilador. Los *scripts* no son tan flexibles en sintaxis y potencia como los programas comunes, pero facilitan las tareas y aumentan la potencia en la creación de páginas *web*.

La misión del siguiente ejemplo consiste en establecer una conferencia a partir de una página *web*. Para ello, será necesario que los miembros de la conferencia tengan instalado el programa *NetMeeting* en su ordenador. Para unirse a una conferencia, pulsaremos el botón **UNIRSE**. Para dejar la conferencia solo habrá que pulsar el botón **DEJAR CONFERENCIA**.

NOTA: *Puesto que debemos asignar un servidor que contenga los usuarios conectados, existe un punto en el programa en el cual hay que cambiar la dirección o nombre del servidor para obtener una conexión.*

En primer lugar definimos la cabecera de la página. En nuestro caso, informamos sobre la utilización de *scripts* para *NetMeeting*:

```
<HTML>
<HEAD><TITLE>Script para NetMeeting:
    Vbsconf</TITLE></HEAD>
<BODY>
<CENTER>
    <h1> VBScript con control ActiveX para
    NetMeeting</h1>
    <h5> [Nota: para utilizar este ejemplo
    debemos cambiar el script
      para utilizar el nombre (o dirección) de
    la máquina que alberga la conferencia]
    </h5>
    <p> Pulsa el botón UNIRSE para
    comenzar! </p>
    <p>
    <p>
    <p>
</CENTER>
<BR CLEAR = left>
```

Tenemos que definir un cuadro de listas (*MemberList*) que contendrá todos los usuarios a los que podemos unirnos en una conferencia. Además, insertaremos dos botones para unirnos o salir de la conferencia actualen la que nos encontramos.

El botón **UNIRSE** se llamará *CallUser* y el botón **DEJAR CONFERENCIA** se llamará *LeaveConf*:

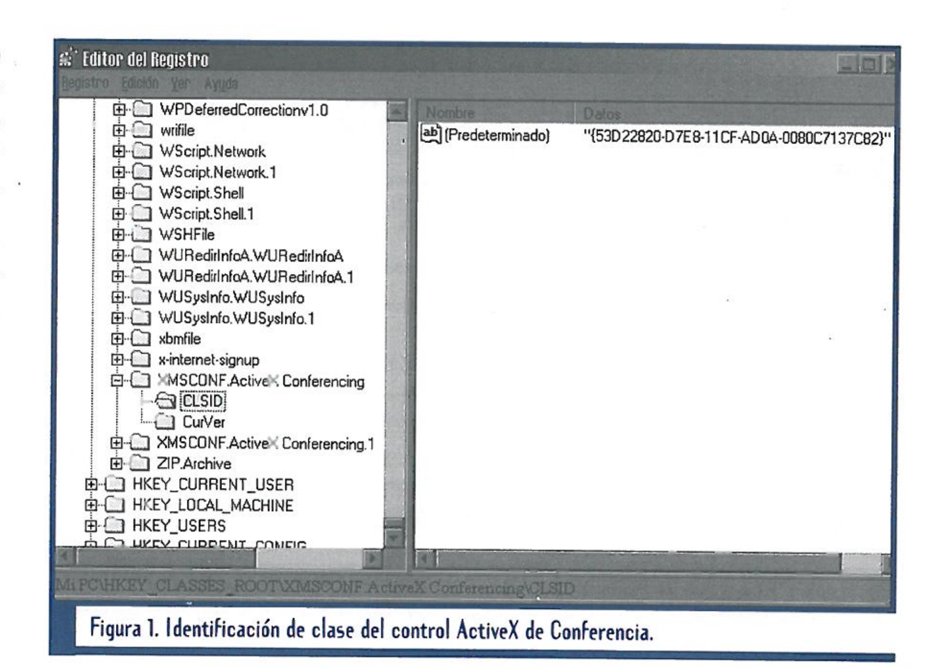


Figura 1. Identificación de clase del control ActiveX de Conferencia.

```
<CENTER>
  <b>Members </b>
  <PRE><TEXTAREA NAME=MemberList
      COLS = 60
   ROWS=5></TEXTAREA></PRE>
  <PRE><INPUT NAME=CallUser
      TYPE=BUTTON VALUE="UNIR-
      SE"><INPUT
   NAME=LeaveConf TYPE=BUTTON
      VALUE="DEJAR CONFEREN-
      CIA"></PRE>
</CENTER>
```

Si utilizásemos el compilador de *Visual Basic* deberíamos insertar el control *ConfMgr* en un formulario para utilizar los recursos de conferencia en *NetMeeting*. De igual modo, tenemos que insertar este control en nuestra página *web*, aunque lo haremos de una forma no visual. Debemos indicar el *ID* y su *CLASSID*.

La tecnología multipunto permite varias conferencias a la misma vez

Esta clase se instala en el registro cuando instalamos *NetMeeting*.

```
<OBJECT
    ID=ConfMgr
    CLASSID="clsid:53D22820-D7E8-11CF-
    AD0A-0080C7137C82">
</OBJECT>
```

Seleccionamos el tipo de lenguaje *script* a utilizar, que en nuestro caso concreto será *Visual Basic*, aunque también es posible utilizar *JavaScript*.

```
<SCRIPT LANGUAGE="VBScript">
Option Explicit

Dim Conference
```

Definimos el primer procedimiento que se ejecutará cuando se cargue la página *web* en el explorador (*Internet explorer 3.0* o superior):

```
Sub Window_OnLoad
   Dim strCaps
   Dim lCaps
   Dim Conferences

   Set Conference = Nothing

   On Error Resume Next
```

Lo primero es inicializar el control *ActiveX ConfMgr*, para lo que utilizaremos el método *Initialize* es:

```
HRESULT Initialize(BSTR bstrID, BOOLEAN
      *pfSucceeded);
```

- *Member*. Participante en una conferencia.
- *Channel*. Conducto para un tipo específico de medio.
- *Audio Channel*. Canal de audio.
- *Video Channel*. Canal de vídeo.
- *File Transfer Channel*. Gestiona la transferencia de ficheros.
- *Application Sharing Channel*. Gestiona las aplicaciones compartidas.
- *Data Channel*. Canal para enviar y recibir datos.

Detallamos los objetos *COM* .

CONFERENCE MANAGER

Este objeto es el encargado de manejar todo el sistema de conferencia. Un objeto **INmManager** se puede crear utilizando *CoCreateInstance* mediante un identificador de clase predefinido (*CLSID_NmManager*) y una interfaz (*IID_INmManager*).

LOCAL SYSTEM

Controla la información sobre el ordenador y usuario local.

CALL

Un objeto de tipo *Call* puede ser instanciado mediante el método *CreateCall* y puede ser creado por el sistema para las llamadas entrantes.

La incorporación de NetMeeting en una página web permite una interacción muy alta

CONFERENCE

Gestiona toda la conferencia. Este objeto es instanciado con el método *CreateConference*. Una aplicación puede instanciar cualquier número de objetos *Conference*, pero sólo uno de ellos puede estar activo al mismo tiempo. Mediante el uso de la interfaz *IEnumNMConference* navegaremos a través de los objetos **Conference** actualmente instanciados.

MEMBER

El objeto *Member* contiene información sobre un participante en una conferencia. Las aplicaciones recibirán notificaciones a través del método *MemberChanged* de la interfaz *InmConferenceNotify* al crearse un objeto *Member*.

CHANNEL

Este objeto es utilizado para la manipulación de los canales que se empleen en una conferencia. Los objetos *Application Sharing, Audio, Data, File Transfer,* y *Video Channel* heredan propiedades y métodos del objeto *Channel*.

DATA CHANNEL

Este objeto, contiene funciones específicas para la manipulación de los canales de datos. *Data Channel* es instanciado con el método *InmConference::CreateDataChannel*. Una aplicación puede crear múltiples *Data Channel*.

AUDIO CHANNEL

Este objeto es derivado del objeto *Channel*. El *Audio Channel* controla el envío y recepción de audio entre el ordenador local y uno de los otros ordenadores de los participantes en la conferencia. Al igual que en la interfaz de usuario de *NetMeeting*, el audio y vídeo pueden intercambiarse entre los diferentes miembros.

VIDEO CHANNEL

Este objeto, el cual es derivado del objeto *Channel*, contiene funciones específicas para manipular los canales de vídeo. Sólo puede haber un canal de vídeo entrante y saliente al tiempo.

FILE TRANSFER CHANNEL

Este objeto, contiene funciones específicas para la manipulación de los canales de transferencia de archivos. Para obtener un objeto *File Transfer Channel*, utilizamos *INmChannel::QueryInterface* con el identificador de Interface *IID_InmChannelFt*. ésta implementa:

- **InmChannelFt Interface**
- **InmChannelFtNotify Interface**

FILE

El objeto *File* maneja información sobre el fichero que se esté transfiriendo y es instanciado mediante el método *INmChannelFt::SendFile*.

APPLICATION SHARING CHANNEL

Este objeto contiene funciones específicas para manipular los canales encargados de compartir aplicaciones entre participantes.

UTILIZANDO VIDEO

En los apartados siguientes vemos los conceptos que debemos tener claros.

- Para utilizar las características de vídeo de *NetMeeting* necesitaremos una tarjeta capturadora de vídeo y una cámara o una videocámara que se conecte a través del puerto paralelo del ordenador .
- El vídeo sólo es soportado sobre protocolo *TCP/IP* y con una sola persona al mismo tiempo.
- La reproducción de vídeo en una

conferencia multiusuario puede impactar negativamente en el rendimiento de todos los ordenadores de la conferencia.

- Si tenemos más de un dispositivo de vídeo instalado, o no se ha desinstalado un dispositivo de vídeo previamente instalado, puede que no seamos capaces de recibir un sólo cuadro de imagen.
- Ordenadores con puertos paralelos bidireccionales funcionarán mucho mejor con cámaras de vídeo sobre puerto paralelo. .
- Las cámaras que se conecten a una tarjeta capturadora de vídeo utilizarán menos recursos del procesador que las conectadas al puerto paralelo. Por ello, es recomendable no utilizar cámaras a color para puerto paralelo si no se tiene un *Pentium 233* o superior.
- Si desconectamos la cámara mientras utilizamos las características de vídeo de *NetMeeting*, el software de nuestra cámara puede que visualice mensajes advirtiéndonos que la cámara no responde.

La utilización de vídeo y audio en una conferencia requiere un Pentium II de última generación

- Se recomienda no ejecutar otro programa que utilice vídeo mientras se esté ejecutando *NetMeeting*.
- Si algunas áreas de nuestra ventana de vídeo contienen falsos colores en la imagen, puede que llegue poca luz a la cámara.
- Si utilizamos características de vídeo en un área oscura, algunas cámaras provocarán que el ordenador se vuelva lento y no responda .
- En algunas cámaras podemos ser capaces de reducir el consumo de *CPU* ajustando manualmente valores en las cajas de diálogo *FUENTE* y *FORMATO* en lugar de dejar que el *driver* de vídeo lo haga .

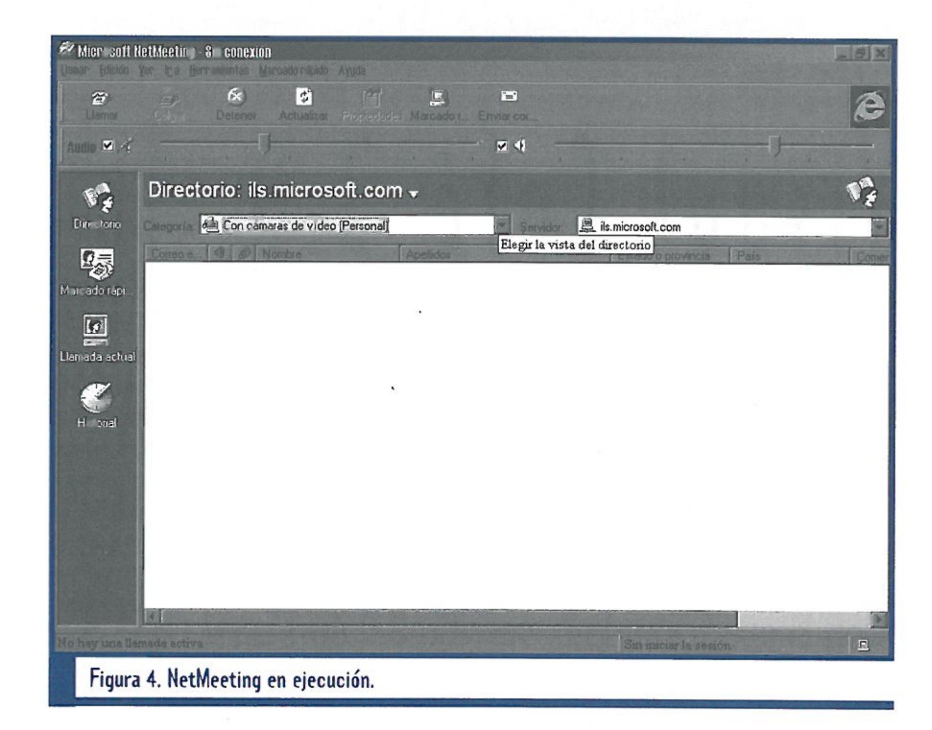


Figura 4. NetMeeting en ejecución.

UTILIZANDO AUDIO

A continuación veremos algunas consideraciones referentes al uso del audio en *NetMeeting*:

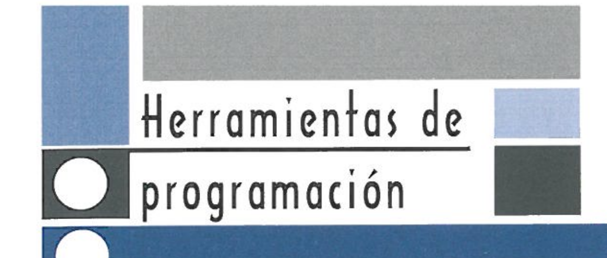
- El audio es soportado sólo sobre protocolo *TCP/IP* y con una persona al mismo tiempo.
- La calidad del sonido puede variar dependiendo de nuestra tarjeta de sonido y el micrófono utilizado.
- El uso de audio *full-duplex* puede requerir más ancho de banda y más *CPU* que si se utiliza *half-duplex*, .
- Si modificamos el *driver* de dispositivo de nuestra tarjeta de sonido, necesitaremos ejecutar el asistente de audio para que *NetMeeting* vuelva a ajustar todos los parámetros correctamente.
- Si experimentamos una pobre calidad de sonido en modo *full-duplex*, asignaremos a *NetMeeting* el modo *half-duplex*. Para hacer esto, debemos seguir los siguientes pasos:

1. Asegurarnos que no estamos en una llamada de *NetMeeting*.
2. En el menú **Herramientas** pulsaremos sobre **Opciones**.
3. En la pestaña *Audio* desactivamos la opción *Full Duplex*.

Es recomendable no cambiar entre *full-duplex* y *half-duplex* mientras estamos en una conferencia.

- Si nuestro ordenador tiene más de un dispositivo de audio, debemos cerciorarnos de que los dispositivos seleccionados en el asistente de audio se corresponden con las propiedades multimedia del *Panel de Control*.
- Los ordenadores con procesadores *486* no serán capaces de utilizar audio con otro software basado en el estándar *H.323* sobre conexiones por módem. Sí podrán hacerlo sobre red de área local (*LAN*).

En el siguiente artículo veremos nuevas aplicaciones que hacen uso de la *API NetMeeting* pero con la utilización de algunos de los objetos *COM* descritos anteriormente en lugar del control *ActiveX*.



Visual Café 3.0, la última herramienta RAD para Java

Javier Sanz Alamillo (jsanza@teleline.es)

Con la aparición de la versión 3.0 de *Visual Café*, *Symantec* ofrece un producto de tercera generación de una gran madurez y se lanza a la carrera por seguir siendo el número uno en ventas de entornos *Java* evolucionados y competitivos.

INTRODUCCIÓN

Con el lanzamiento del nuevo *Visual Café* versión 3.0, *Symantec* se consolida como uno de los principales fabricantes de herramientas para *Java* que tener en cuenta. Con la presentación de tres diferentes ediciones de este producto, que se pueden escoger en función de las necesidades requeridas, se cubre un amplio abanico de posibilidades, desde el puramente académico mediante el cual se puede introducir a los nuevos programadores el lenguaje *Java* hasta un entorno que permite el desarrollo de potentes aplicaciones distribuidas con accesos a bases de datos, etc. No debemos olvidar que se han añadido nuevas características, se han mejoras considerables y también correcciones sobre las posibilidades de versiones anteriores.

Visual Café es un completo entorno *RAD* (*Rapid Application Development*) diseñado exclusivamente para el desarrollo de aplicaciones *Java*. Es un entorno del tipo *WYSIWYG* (*What you see is what you get*, "lo que ve es lo que consigue"), con un gran conjunto de componentes mejorados y ampliados, con la posibilidad de realizar un cómodo *debug* y desarrollar aplicaciones *stand-alone* y *applets* de alto rendimiento. Con las herramientas para manejar *JavaBeans*, ficheros *Jar* y bases de datos, nos encontramos ante un entorno de desarrollo flexible, potente y práctico.

INSTALACIÓN

La instalación de *Visual Café* es un proceso sencillo, similar en todas las ediciones, que sigue la misma línea de instalación que cualquier otro tipo de software, esto es, seguir las instrucciones del asistente.

Respecto a las características del ordenador en el cual vamos a realizar la instalación de *Visual Café* en cualquiera de sus ediciones, los únicos requisitos a tener en cuenta antes de realizar la instalación, con independencia de que *Visual Café* está diseñado para funcionar bajo entorno *Windows*, son el espacio en disco requerido y la memoria RAM.

Respecto al espacio en disco, ronda entre los 160 Mb y 238 Mb, en función del tipo de sistemas de ficheros que estemos utilizando, como *FAT-16*, *FAT-32* o *NTFS*, y del tamaño del *cluster* de disco. Esto viene a remediar una queja constante en los usuarios que indican repetidas veces que es el excesivo el tamaño de disco duro que requiere *Visual Café*.

En la parte que hace referencia a la memoria *RAM* necesaria para trabajar con *Visual Café* normalmente, siguiendo la tónica común en todos los entornos *RAD* de Java, siempre se recomiendan 48 Mb, pero la experiencia demuestra que 64 Mb a veces resultan demasiado justas. Disponer de 96 Mb o más hace que se pueda utilizar el entorno con toda

la potencia y flexibilidad que ofrece. Si bien esta versión requiere más o menos los mismos recursos que las anteriores, se observa como la velocidad de trabajo se ha mejorado, de tal forma que el proceso de desarrollo resulta más cómodo y productivo. Otras recomendaciones:

- No instale *Visual Café 3.0* en los mismos directorios de versiones anteriores. Aunque el producto toma precauciones de no utilizar un nombre de directorio de versiones anteriores, evite cambiar el nombre del directorio de instalación.
- No es necesario desinstalar versiones anteriores de *Visual Café* para utilizar la versión 3.0.
- Y como anécdota curiosa, no utilice el símbolo % en el nombre de un directorio, puesto que produce confusiones varias en la localización de los archivos. Aunque no lo parezca, *Symantec* recibió reclamaciones por esta circunstancia.

CARACTERÍSTICAS GENERALES DE VISUAL CAFÉ 3.0

Visual Café 3.0 ofrece a los desarrolladores poder elegir entre tres diferentes ediciones del producto, en función de sus necesidades, cada una de las cuales consta de características propias. A continuación se indican las diferencias fundamentales entre las distintas ediciones:

- *Estándar*. Entorno de desarrollo que facilita el desarrollo de *applets* orientados al *Web* y otras sofisticadas aplicaciones. Es ideal para programadores que estén iniciándose en *Java* y para aquellos que empiezan a usar un entorno *RAD*, por su fácil uso y rápido aprendizaje.
- *Profesional.* Indicada para los desarrolladores experimentados que necesitan utilizar las últimas novedades en *Java* y desean crear aplicaciones potentes. Incluye todas las opciones de la edición estándar más nuevas posibilidades.
- *DataBase u orientada a bases de datos*. Se trata del entorno ideal para el desarrollo de aplicaciones que incluyan bases de datos y opciones de operación en entornos distribuidos. Incluye las características de la edición profesional.

La aparición del JDK 1.2 ha supuesto la creación de nuevos entornos RAD para Java

El uso de *Visual Café* en cualquiera de sus ediciones supone un incremento en las posibilidades de los desarrolladores, tanto para los que están empezando a utilizar *Java* como para los que ya han realizado programas con alguna versión de este producto. A continuación se van a detallar las características más importantes que ofrece *Visual Café 3.0* en sus distintas ediciones.

- *JDK 1.1.7a y JDK 1.2.*

  Visual Café utiliza la versión *1.1.7a* del *JDK* (*Java Development Kit*). Mediante opciones de menú disponibles se puede cambiar esta opción y así poder utilizar la versión *1.2* del *JDK* ofrecida por *Sun*. Conviene indicar que el uso del *JDK* propio de *Sun* hará que el entorno se ralentice, debido a que esa máquina virtual es algo más lenta que la ofrecida en el entorno y algunas de las opciones disponibles serán limitadas o anuladas. Las incompatibilidades descritas en la versión del *JDK 1.1.7* por *Sun* son idénticamente aplicables en este entorno.

- *Swing*

  Visual Café incluye la implementación completa de *JFC/Swing* versión *1.0.3*. En anteriores versiones constaba de algunos componentes puestos en funcionamiento de forma parcial. En esta última, se pueden construir entornos gráficos en los que gestionar su apariencia y el comportamiento no supone un gran esfuerzo de programación, con la ventaja de disponer de una interfaz gráfica que requiere menos recursos que si se usara el clásico *AWT*. Debido a que el programador puede disponer de varias implementaciones de *Swing*, no es recomendable cambiar el fichero **swingall.jar** incluido en *Visual Café 3.0* por otro disponible. Para completar el manejo de *Swing* se ha incluido la opción **Swing Menu Designer** que hace más fácil crear o cambiar menús.

- Asistentes

  Se ha mejorado e incrementado el conjunto de asistentes. Los cambios pasan por avances en los distintos pasos para construir una aplicación, cambios en las presentaciones de las diferentes opciones, etc., todo ello para conseguir que el proceso resulte claro y conciso.

- *Javadoc*

  Gracias a las mejoras presentadas en la herramienta *Javadoc*, ahora se pueden crear y modificar los comentarios generados de forma visual gracias a un nuevo editor, construyéndose así fácilmente los ficheros *HTML* correspondientes. Indicar que los usuarios de *Windows 98* tendrán ciertos problemas con los directorios que contienen las imágenes de las paginas generadas.

- Mejoras en los ficheros *Jar*

  Aunque se ha cambiado el nombre de la opción para la gestión de ficheros *jar*, que ahora se denomina *Deploy,* se siguen manteniendo las anteriores posibilidades y se han añadido algunas nuevas. Por ejemplo, la actualización de ficheros contenidos en un *jar* no obligará a crear otro fichero *jar* nuevo. Automáticamente se generará uno con los nuevos cambios, siendo este proceso transparente para el usuario.

También se pueden crear ficheros *ZIP* y *CAB* mediante una mínima configuración en una de las opciones de

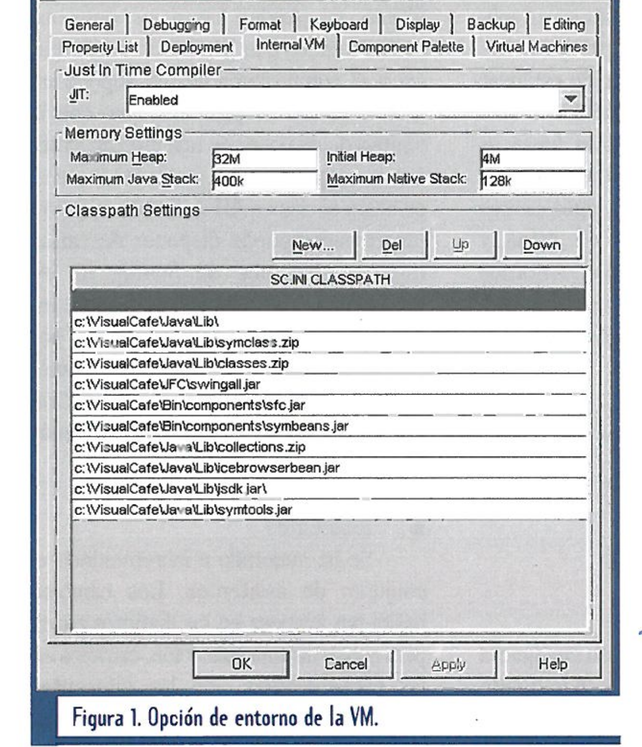


Figura 1. Opción de entorno de la VM.

menú, del mismo modo que anteriormente se gestionaban los ficheros *jar*, pudiéndose usar igualmente la herramienta *AutoJar* y *JAR viewer*. Se ha añadido una opción de manejo de ficheros *jar* mediante *ftp*, en la que añadir nuevos *jar* pasa por indicar la dirección de *ftp* y descargarlos, configurándose de forma automática.

Visual Café 3.0 se presenta en tres ediciones diferentes

- Opciones de entorno de la *VM*

Ahora ya no es necesario modificar manualmente el fichero *sc.ini* para indicar los directorios en los que buscar los ficheros de clases. Se ha añadido una nueva opción que permite realizar esta tarea fácilmente, además de permitir indicar características de ejecución de la aplicación, como la memoria inicial..

- Eventos e interacciones

Esta última versión de *Visual Café* incluye un sistema de desarrollo de aplicaciones que se denomina de eventos e interacciones, similar en funcionamiento y desarrollo al disponible y utilizado por los programadores que utilizan el producto *VisualAge* de *IBM*. Principalmente se basa en definir relaciones causa-efecto entre los componentes que se estén utilizando, definiendo así el desarrollo de la aplicación. Es un método muy fácil de aprender y utilizar, por lo que los desarrollos suelen ser muy rápidos y cómodos.

- Opciones *Java Code Helper y Two-ways drag and drop*

Mediante la nueva opción **Code Helper** el programador encontrará un ayudante que le indicará por ejemplo, qué parámetros debe definir para llamar a una función determinada si se ha olvidado, o si una variable pertenece a una determinada clase u otra, etc.

Mediante el sistema *two-ways drag and drop*, se define un sistema de desarrollo en el cual se pueden realizar cambios en una aplicación de forma que se puede observar este cambio tanto en el ámbito visual como el propiamente generado como código.

- Localización

Ahora se dispone de una serie de herramientas y asistentes para que todo el proceso de localización de una aplicación sea más sencillo de realizar. Gracias a ello, cualquier cambio o modificación pasa por usar las opciones de menú disponibles.

- *Live Update*

Puesto que surgen constantemente actualizaciones, cambios y modificaciones en todo lo relacionado con Java, *Visual Café* ofrece la posibilidad de realizar una actualización del producto mediante el cambio de determinadas partes que *Symantec* ofrezca.

DESCRIPCIÓN DE LAS DIFERENTES EDICIONES

A continuación se van a detallar las características fundamentales de las distintas ediciones que permitirá al desarrollador escoger el entorno de desarrollo que mejor se adapta a sus requerimientos y necesidades.

VISUAL CAFÉ STANDARD EDITION 3.0

Como se comentó anteriormente, esta edición resulta ideal para aquellos que comienzan a desarrollar con *Java*, o para los que cuentan con unos conocimientos básicos y desean un entorno *RAD* de fácil uso y aprendizaje, sin que esto sea excluyente para disponer de un entorno potente y rápido.

Incluye un entorno *IDDE* (*Integrated Development and Debugging Environment*) en el que se pueden diseñar *applets* y aplicaciones sin tener que escribir una línea de código. Es un entorno totalmente integrado, puesto que se puede compilar una aplicación y ejecutarla sin necesidad de realizar cambios extraños o salir del mismo. Incluye un gran número de asistentes, utilidades y una librería con más de cien *JavaBeans* con el código fuente disponible, que permitirá al desarrollador a generar sus propios *beans* o mejorar cualquiera de los existentes. *Visual Café* presenta la opción *two-way drag and drop*, que como se detalló anteriormente, consiste en que el programador puede escribir su código en el editor, usar el diseño visual o utilizar los dos

a la vez, permitiendo la creación de código que se actualiza en tiempo real. Además, se incluye el soporte para la versión *JDK 1.2* y la última tecnología sobre *JavaBeans*, *Jars*, *Jini*, serialización, etc.

Se han aumentado los ficheros de ayuda disponibles para el desarrollador, incluyéndose gran cantidad de ejemplos en muchas opciones que aparentemente parecen oscuras para los programadores. Una vez que el programador se familiariza con el entorno, comprobará como incrementa su facilidad a la hora de crear nuevos desarrollos gracias al conjunto de herramientas disponibles.

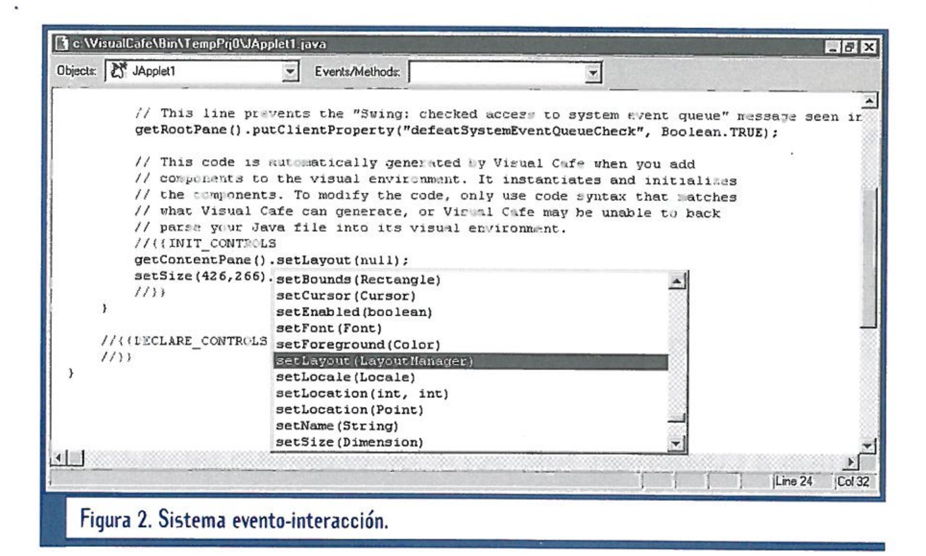


Figura 2. Sistema evento-interacción.

VISUAL CAFÉ PROFESSIONAL EDITION 3.0

Incluye todas las posibilidades de la edición *Standard* más un extenso conjunto de *JavaBeans* que facilitará el desarrollo de *beans* similares. Incluye un conjunto de asistentes para el desarrollo de *Servlets* y tareas de localización.

Visual Café utiliza la versión 1.1.7a del JDK

Dispone de la nueva herramienta *Java Code Helper*, que como se detalló, ayuda al programador a escribir código más seguro, indicándole según lo genera por ejemplo, los parámetros correspondientes a una función a la que esta invocando, evitando así que se pasen parámetros erróneos, etc. Las posibilidades de *debugging* se han mejorado e incrementado. Se puede trazar código de *applets* en local, *applets* de *web*, aplicaciones, *servlets*, *debug* remoto, etc.

Si bien *Visual Café* incluye el *JIT* más rápido del mercado en todas sus ediciones y uno de los mejores generadores de código *x86*, éste último se ha optimizado un poco más, por lo que las aplicaciones generadas en código nativo incrementan su velocidad en un 45%, haciéndolas competitivas. Con la nueva versión de *Version Control Bridge (VCB)*, se puede integrar el sistema de control de versiones que utilice con *Visual Café*, ya que permite utilizar *software* de control de versiones de terceras partes para gestionar los cambios en los desarrollos, manteniendo las versiones correctas fuera de todo riesgo de cambio y permitiendo el control de acceso a determinado código.

Visual Café soporta dos versiones:

- *Visual Café Version Control*, que permite a los vendedores de *software* de control de versiones integrar su producto mediante una serie de *plug-ins*.
- La interfaz *Microsoft Source Code Control(SCC)* es soportada por *Visual Café*, por lo que se puede utilizar todo el *software* de control que sigue esta interfaz.

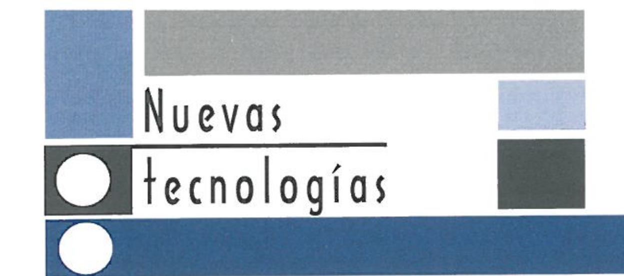
VISUAL CAFÉ DATABASE EDITION 3.0

Con este producto se dispone de todas las posibilidades de la edición Profesional, añadiendo mejoras en *JFC/Swing*, actualización y nuevas posibilidades en la gestión de base de datos y lo que es más novedoso, permite la gestión de *drivers JDBC*. Consta de asistentes para la creación de componentes, la gestión de la localización y para el desarrollo de aplicaciones que utilicen bases de datos. Se ha incluido la posibilidad de utilizar *SQL* mediante el uso de *Query By Examples* (*QUE*) que permite a los usuarios construir consultas *SQL* sin conocimientos de bases de datos. Todo un conjunto de añadidos que hacen que esta opción sea la ideal para el desarrollo de aplicaciones con bases de datos en *intranets*, etc.

Debido al gran número de opciones, posibilidades y mejoras, cualquier desarrollador puede encontrar su entorno ideal.

CONCLUSIÓN

Como se ha podido comprobar a lo largo de este artículo, *Visual Café 3.0* supone una gran apuesta de *Symantec* para hacerse con el número uno en los entornos *RAD* para *Java*. Gracias a la madurez de este producto, a sus actualizaciones, mejoras y ampliaciones, se dispone de un entorno ideal de desarrollo, tanto para el que desea iniciarse como para el que tiene que afrontar el desarrollo de una aplicación importante.



DirectX 6.1 (I)

Constantino Sánchez Ballesteros (constantino@nexo.es)

Vamos a comenzar una serie de artículos basados en la ultimísima versión de las librerías *DirectX* (creadas por *Microsoft* principalmente para desarrollar contenido multimedia avanzado y videojuegos) para tomar contacto con las nuevas opciones incluidas en esta tecnología puntera.

INTRODUCCIÓN

Desde la aparición en el mercado de las librerías *DirectX* de *Microsoft*, la programación de vídeojuegos bajo Windows cambió radicalmente. Pocos programas se dirigen ya hacia el paleolítico *MS-DOS*, puesto que la tecnología avanza a pasos agigantados y este sufrido sistema operativo tiene cada vez menos soporte para el nuevo hardware que se va implementando.

Como ejemplo, hay que destacar las nuevas aceleradoras *3D*, ya que cualquier juego bajo *MS-DOS* que quiera utilizar las posibilidades de estas extraordinarias tarjetas gráficas necesita ser programado exclusivamente para cada una de ellas, es decir, que la programación efectuada en relación con una tarjeta no funcionará con otra de distinto fabricante. Este hecho no ocurre bajo *Windows 98*, puesto que cada tarjeta implementa soporte para las librerías *DirectX* y el programador sólo tiene que centrarse en esta *API*, sin importarle el tipo o clase de tarjeta gráfica que tenga instalada el PC.

De este modo, se evita el caos gráfico que existía en *MS-DOS* cuando aparecieron las tarjetas *SVGA* y su compatibilidad con el estándar *VESA*. *Windows* se presenta realmente como la plataforma ideal para la creación de videojuegos, puesto que todo el *hardware* que tenga instalado nuestro *PC* estará registrado en el sistema operativo, y si funciona con él, también funcionará cuando utilicemos *DirectX*. La plataforma *DirectX* es un conjunto de *API's* que permite a los desarrolladores de contenido interactivo acceder a características de *hardware* especializado sin tener que escribir código específico de hardware.

Para estos desarrolladores, *DirectX* ofrece la estabilidad y la estandarización en un mercado imprevisible y cambiante. Para los usuarios de juegos y otros contenidos, el resultado es un mayor realismo e interactividad y una mayor selección de títulos compatibles con una amplia gama de hardware. Con estos detalles comentados se puede decir que *DirectX* se sitúa como el estándar multimedia interactivo en la industria de los PC's, con un grado de madurez considerable.

DirectMusic es el nuevo componente musical incluido en el SDK de DirectX

Recientemente ha aparecido la versión 6.1 de esta *API* con interesantes novedades. *DirectX 6* abre una nueva dimensión en el entretenimiento al permitir los juegos *online* con varios jugadores, sonido en tres dimensiones y soporte para nuevos tipos de dispositivos y periféricos. Una ventaja clave de *DirectX* reside en la integración de una amplia gama de *API's* necesarias para *Internet* y contenido interactivo en *CD-ROM*. Las facetas clave para la creación de contenido multimedia son gráficos en 2D y 3D, vídeo, sonido, música, dispositivos de entrada y conectividad.

COMPONENTES DE DIRECTX

El nuevo *SDK* de *DirectX* incluye una novedad en relación a la versión anterior (6.0); *DirectMusic*. En esta sección se listan todos los componentes incluidos en el *SDK* ofreciendo una breve descripción de sus posibilidades.

- *DirectDraw*: acelera las técnicas de animación mediante *hardware* o *software* permitiendo acceso directo a los *bitmaps* albergados en la memoria de vídeo así como capacidades de *flipping-buffering* (volcado rápido de zonas de memoria que contienen gráficos).
- *DirectSound*: permite mezcla de sonido y reproducción por *hardware* y *software*.
- *DirectMusic*: es el nuevo componente de *DirectX*. Se podría definir como el componente musical del *SDK* y a diferencia de *DirectSound* que reproduce *samples* de sonido digital, *DirectMusic* trabaja con datos musicales.
- *DirectPlay*: permite la conexión de juegos sobre un módem o red local.
- *Direct3D*: tiene una interfaz de alto nivel (modo retenido) que permite a los programadores implementar un sistema gráfico 3D completo. También incluye una interfaz de bajo nivel (modo inmediato) que permite a las aplicaciones tomar un control del motor de *Rendering*.
- *DirectInput*: se encarga del manejo de periféricos específicos como el *joystick*, ratón, teclado, y dispositivos *force feedback* (palancas de mando con sensores para efectuar rotaciones y fuerzas en la palanca).
- *DirectSetup*: utilizado para crear nuestros propios programas de instalación.
- *AutoPlay*: es una característica de *Windows* que ejecuta un programa de instalación o juego de forma automática a partir de un *CD-ROM*.

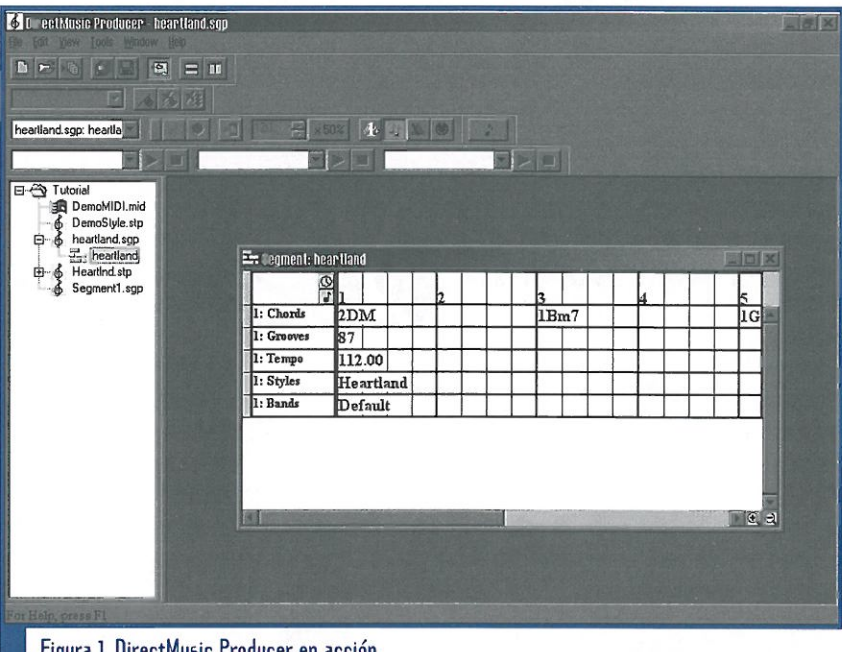


Figura 1. DirectMusic Producer en acción.

NOVEDADES IMPLEMENTADAS

DirectX 6.1 contiene la versión oficial de *DirectMusic* (la nueva *API* musical). Además, se han incluido otras pequeñas características en algún componente. El objetivo de esta sección consiste en ayudar a aquellos programadores familiarizados con versiones anteriores de *DirectX* para que identifiquen rápidamente los nuevos métodos e interfaces implementadas.

DirectX se sitúa como el estándar multimedia interactivo en la industria de los PC's

- **DirectDraw**

DirectDraw introduce las interfaces *IDirectDraw4* e *IDirectDrawSurface4*, que incluyen datos privados de *buffers* y valores únicos para dichos *buffers*. Se han introducido nuevas características en las estructuras *DDCAPS*, además de aparecer *DDSCAPS* y *DDSURFACEDESC2*, ambas utilizadas para definir las propiedades de un buffer gráfico. Las aplicaciones pueden utilizar el nuevo método *TestCooperativeLevel* para determinar el estado de un nivel cooperativo para *DirectDraw*. Esto es útil cuando las aplicaciones necesitan información sobre cuándo se pueden restaurar o recrear los *buffers* que utilicen. Una nueva interfaz, *IdirectDrawGammaControl*, permite a las aplicaciones ajustar de forma sencilla la forma en que se visualizan los gráficos sin cambiar los contenidos *frame buffer*.

- **DirectSound**

No presenta cambios.

- **DirectMusic**

DirectMusic es una nueva *API* que se creó como primicia en el *DirectX 6.0 SDK* y es ya una realidad totalmente funcional en *DirectX 6.1 SDK*.

- **Direct3D Modo Inmediato**

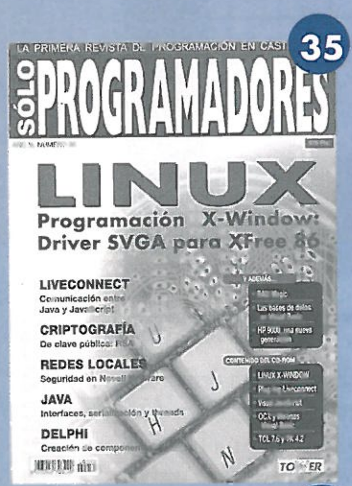
Ahora es más potente que antes, incluyendo soporte para nuevas características *hardware*:

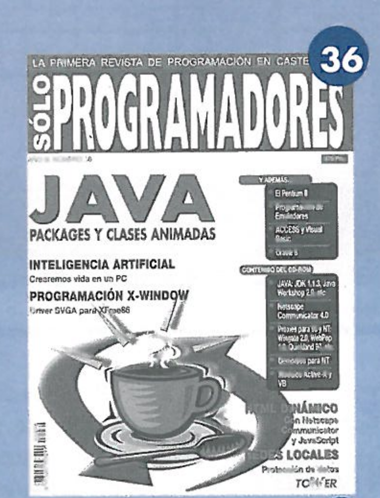
  - Soporte de *single-pass multiple texture blending* (mezcla de múltiples texturas en una pasada de *Render*).

números atrasados

SÓLO PROGRAMADORES

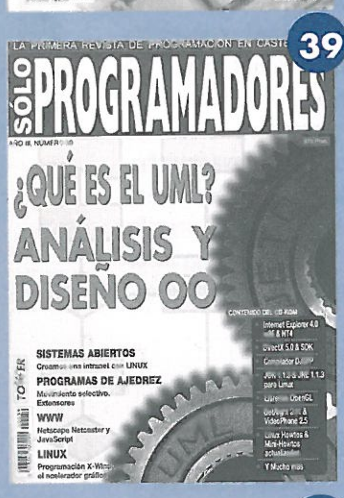
completa ya tu colección

































suscríbete

a Sólo Programadores

y consigue un **magnífico descuento**

suscripción **normal**

ahorro 2.200 ptas.

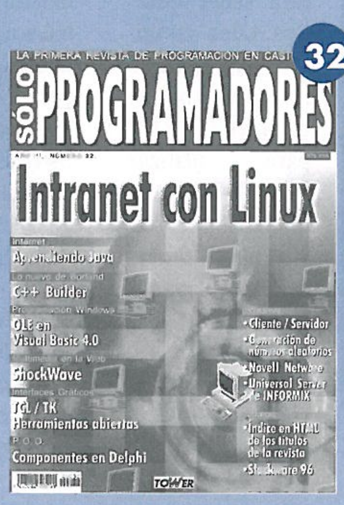
12 revistas (1 año) por sólo... 9.500 ptas.

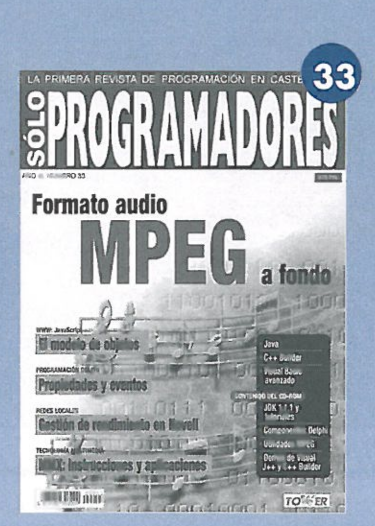
suscripción **estudiantes** (carreras técnicas)

ahorro 4.100 ptas.

12 revistas (1 año) por sólo... 7.600 ptas.









- *Bump mapping* (texturas abultadas dependiendo de la luz).
- Gestión automática de la caché de texturas.
- Formatos de vértices flexibles.
- *Vertex buffers* .
- *W-buffering*.
- *Stencil buffers*.

- **Direct3D Modo Retenido**
  Introduce una nueva interfaz que hereda semánticas e interfaces de otros .

- **DirectInput**
  No presenta cambios.

- **DirectPlay**
  Ofrece una nueva Interfaz *DirectPlay4*, que hereda los métodos de *DirectPlay3*. Toda la nueva funcionalidad es activada a través de nuevos métodos y *flags*. *DirectPlay* soporta estas nuevas características:

  - Mensajes garantizados.
  - Mensajes asíncronos.
  - Soporte *firewall* en *TCP/IP*.
  - Seguridad *peer-to-peer*.
  - Registro de aplicaciones.
  - Soporte de localización.
  - Proveedor de servicios privados.
  - Nuevas características de conexión de *lobby's*.

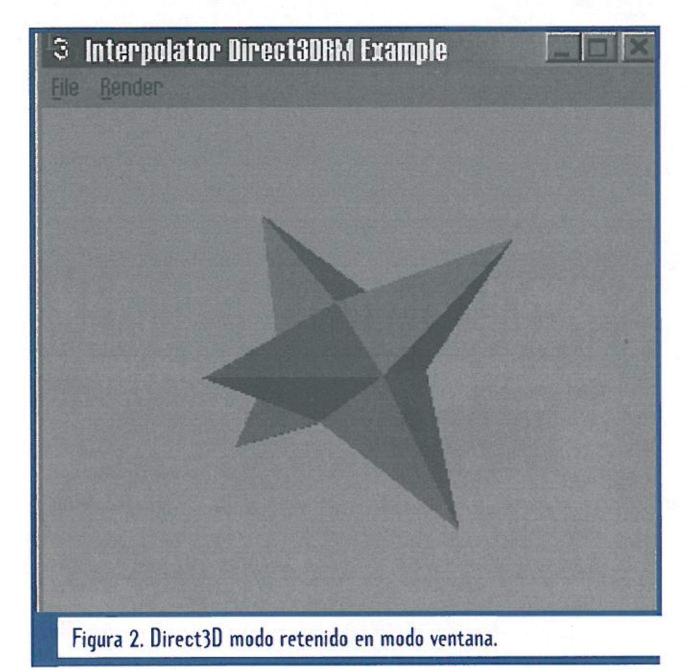


Figura 2. Direct3D modo retenido en modo ventana.

- **DirectSetup**
  No presenta cambios.

- **AutoPlay**
  No presenta cambios.

DIRECTDRAW

DirectDraw es el componente de *DirectX* que permite manipular la memoria de vídeo, el *hardware blitter*, soporte de *overlays* y *flipping surfaces*. Esta no es una *API* de alto nivel para la manipulación de gráficos. *DirectDraw* provee un dispositivo independiente para la creación de juegos y *software Windows* (paquetes gráficos, etc.) ganando acceso a las características de cada tarjeta gráfica específica.

Del mismo modo trabaja con una amplia variedad de *hardware* gráfico, utilizando desde simples *SVGA* hasta las aceleradoras 3D más modernas. La interfaz se ha diseñado para que nuestras aplicaciones puedan enumerar las características del *hardware* y utilizar toda la aceleración que se permita. Aquellas características que no estén implementadas por *hardware* serán emuladas directamente por *software*.

Nuestra aplicación sólo necesitará saber detalles tales como son por ejemplo los formatos de color *RGB* y *YUV* para utilizar *DirectDraw* inmediatamente. No necesitamos realizar llamadas específicas para utilizar el *blitter* o manipular los registros de la paleta gráfica.



DIRECTSOUND

Esta *API* conforma el componente de sonido de *DirectX*. *DirectSound* permite la mezcla de sonidos, aceleración *hardware* y acceso directo al dispositivo de sonido. *DirectSound* también permite captura de sonido y reproducción.

DIRECTMUSIC

DirectMusic es el componente musical de *DirectX*. Realmente, es el nuevo componente incorporado a esta nueva versión de *DirectX (6.1)*. A diferencia de *DirectSound*, que se utiliza para captura y reproducción de *samples* de sonido, *DirectMusic* trabaja directamente con datos musicales que son convertidos en *samples* mediante el sintetizador por *hardware* o *software*. La implementación *software* utiliza el sintetizador *software* de para crear *samples*.

DirectDraw permite el soporte 2D necesario para aplicaciones multimedia y juegos

Además de soportar la entrada de instrumentos musicales en formato *MIDI*, *DirectMusic* puede componer música en tiempo de ejecución. Esta música no es generada de forma algorítmica pero se basa en elementos creados por un compositor humano. Al igual que los demás componentes de *DirectX*, *DirectMusic* es una *API* basada en el modelo de objetos *COM*. La *API DirectMusic* direcciona los fundamentos para crear música del siguiente modo:

- *Reproducción:* utilizando sonidos *DLS*, la música sonará igual en cualquier equipo, además de poder utilizar instrumentos creados por el usuario.

- *Jitter-free timing*. Reproducción de sonido *MIDI* con una precisión de 2 milisegundos.

Además, *DirectMusic* permite importantes características para un fácil desarrollo de composiciones musicales:

- Mecanismo para carga y distribución de segmentos musicales.
- Múltiples rendimientos: más de una pieza musical puede ser reproducida al mismo tiempo, con tempos separados.
- Más de 16 canales *MIDI*. Mapeando canales en grupos de canales, *DirectMusic* sobrepasa la limitación de 16 canales y hace posible la reproducción de un número de voces de forma simultánea, hasta llegar al límite del sintetizador.

Direct3D se compone de dos API's para trabajar a alto y bajo nivel con el hardware

- Manejo automático de instrumentos *DLS*.
- Reproducción dinámica e interactiva: en combinación con *DirectMusic Producer*, el rendimiento del *engine* de *DirectMusic* puede ser utilizado para crear bandas sonoras dinámicas basadas en material musical ya creado.
- Sincronización de toda la reproducción de la música a través del uso de un sólo reloj.

DIRECTPLAY

DirectPlay es una interfaz *software* que simplifica el acceso de las aplicaciones a los servicios de comunicación. *DirectPlay* llega con una tecnología que no sólo provee una forma para que las aplicaciones se comuniquen unas con otras, independientemente del transporte o protocolo, sino que además permite esta independencia para crear servidores de juegos. Las aplicaciones (especialmente juegos) pueden ser más divertidas si pueden ser practicados con jugadores reales y permitiendo un número elevado de ellos.

DIRECT3D

Direct3D se diseñó para permitir gráficos 3D de alto rendimiento. Puede utilizar tanto emulación *software* como las aceleradoras más rápidas y nuevas del mercado. Su misión consiste en ofrecer acceso dependiente del dispositivo a la tarjeta 3D de manera independiente. Dicho de otra manera, *Direct3D* es una interfaz de dibujo para *hardware* 3D. Podemos utilizar *Direct3D* de dos modos bien diferenciados:

1. Modo Retenido.
2. Modo Inmediato.

El modo Retenido es una *API 3D* de alto nivel para programadores que requieren un desarrollo rápido o requieren una fácil construcción de animaciones 3D. *Microsoft* desarrolló el modo inmediato de *Direct3D* como *API* 3D de bajo nivel. Este modo resulta ideal para desarrolladores que necesiten portar juegos y otras aplicaciones multimedia al sistema operativo *Windows*. Se comunica con el *hardware* siempre a bajo nivel, por lo que trabaja más rápido que el modo Retenido. A continuación se exponen algunas de las características soportadas por *Direct3D*:

- *Depth buffering* (con *z-buffers* o *w-buffers*).
- Sombreado *Flat* y *Gouraud*.
- Múltiples luces y tipos de luces.
- Soporte total de materiales y texturas.
- Potentes *drivers* de emulación *software*.
- Transformaciones y *clipping*.
- Independencia *hardware*.
- Soporte total para *Windows NT/Windows 2000*.
- Soporte para al arquitectura *MMX* de *Intel*.

Como se comentó anteriormente, los desarrolladores que quieran diseñar

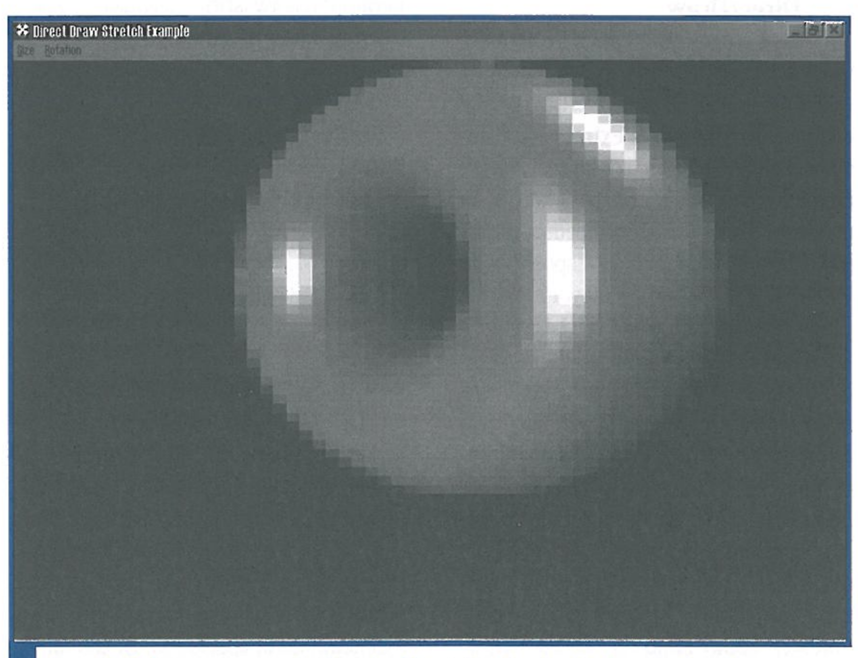


Figura 3. En la imagen se está utilizando el Stretching (ampliación de sprites) mediante DirectDraw

exclusivamente juegos, deberán inclinarse por la utilización del modo Inmediato de *Direct3D*, aunque ello conlleve más quebraderos de cabeza y conocimientos que el modo Retenido. Se trata de la *API* diseñada para el control de periféricos tales como el ratón, teclado, *joysticks*, etc.

PROGRAMACIÓN DE DIRECTDRAW

En el siguiente ejemplo aprenderemos a programar nuestra primera aplicación con *DirectDraw* utilizando el compilador *Visual C++* de *Microsoft*. Es necesario que en el entorno del compilador se referencien todas las librerías necesarias para compilar una aplicación que utilice *DirectX*. En la estructura de directorios de búsqueda del programa podréis asignar los *paths* que contienen dichas librerías y *headers*. El programa visualizará alternativamente texto informándonos del *buffer* que se está visualizando.

- **Fase 1: creación del objeto DirectDraw**

Para crear una instancia de un objeto *DirectDraw* nuestra aplicación debería utilizar la función *DirectDrawCreate* tal y como se muestra en el extracto de listado presentado más adelante. *DirectDrawCreate* contiene tres parámetros. El primero toma un identificador global *(GUID)* que representa el dispositivo de vídeo. El *GUID*, en la mayoría de los casos, se establece como **NULL** y significa que *DirectDraw* utilizará el *driver* de la tarjeta de vídeo por defecto que esté instalado en el sistema. El segundo parámetro contiene la dirección de un puntero que identifica la localización del objeto *DirectDraw* si es que fue creado. El tercer parámetro siempre se establece como **NULL** y se ha incluido para futuras expansiones. El siguiente ejemplo muestra cómo crear el objeto *DirectDraw* y cómo determinar si la creación se estableció correctamente o no:

```
ddrval = DirectDrawCreate(NULL, &lpDD,
      NULL);
if(ddrval == DD_OK) {
  // lpDD es un objeto DirectDraw válido.
}
else {
  // El objeto DirectDraw no pudo ser creado.
}
```

- **Fase 2: determinando solicitudes de la aplicación**

Antes de que podamos cambiar la resolución del modo de vídeo debemos especificar unos *flags* mínimos *(DDSCL_EXCLUSIVE* y *DDSCL_FULLSCREEN)* en el parámetro *dwFlags* del método *SetCooperativeLevel*. Esto da a nuestra aplicación un control completo sobre el dispositivo de vídeo, y ninguna otra aplicación será capaz de compartirlo. Además, el *flag DDSCL_FULLSCREEN* pone la aplicación en modo exclusivo (pantalla completa). Esto significa que nuestra aplicación cubre todo el escritorio y sólo nuestro programa puede escribir en pantalla. El siguiente fragmento demuestra el uso del método *SetCooperativeLevel*:

```
HRESULT      ddrval;
LPDIRECTDRAW lpDD;   // creado anterior-
      mente por DirectDrawCreate
 ddrval = lpDD->SetCooperativeLevel(hwnd,
      DDSCL_EXCLUSIVE |
   DDSCL_FULLSCREEN);
if(ddrval == DD_OK) {
  // se estableció el modo exclusivo
}
else {
  // no se pudo asignar el modo exclusivo
}
```

Si *SetCooperativeLevel* no devuelve *DD_OK*, aún podremos ejecutar nuestra aplicación. El programa no estará en modo exclusivo y puede que no obtenga el rendimiento que debiera de nuestra tarjeta de vídeo. En ese caso podríamos visualizar un mensaje que permitiera al usuario continuar o no la ejecución de la aplicación. Si activamos el modo pantalla completa deberemos pasar a nuestra aplicación un *handle* en *SetCooperativeLevel* para permitir a *Windows* determinar si nuestra aplicación termina de forma anormal.

El componente DirectSound no presenta cambios

- **Fase 3: cambiando el modo de vídeo**

Después de que establecemos las solicitudes de la aplicación, podemos utilizar el método *SetDisplayMode* para cambiar la resolución del modo de vídeo. El siguiente ejemplo muestra cómo establecer el modo de vídeo a 640x480x8 bpp (256 colores):

```
HRESULT      ddrval;
LPDIRECTDRAW lpDD;  // creado anterior-
      mente.
 ddrval = lpDD->SetDisplayMode(640, 480, 8);
if(ddrval == DD_OK) {
  // modo de vídeo cambiado satisfactoriamente.
}
else {
  // el modo de vídeo no se pudo establecer
}
```

Al establecemos el modo de vídeo, debemos asegurarnos que si el *hardware* del usuario no puede soportar altas resoluciones de pantalla, nuestra aplicación se establecería en un modo de vídeo estándar que no diera problemas. Por ejemplo, 640x480x800 es un valor estándar típico para cualquier tarjeta de vídeo.

- **Fase 4: creando Flipping Surfaces**

Después de establecer el modo de vídeo, debemos crear los *buffers* que utilizará nuestra aplicación. Para realizar el *Flipping buffering*, debemos crear dos *buffers* en memoria, uno primario y otro denominado *backbuffer*. El primer paso consiste definir las propiedades del *buffer* que se utilizará como primario y representará la pantalla:

```
// Creamos el primario con un backbuffer
ddsd.dwSize = sizeof(ddsd);
```

```
ddsd.dwFlags = DDSD_CAPS | DDSD_BACK-
        BUFFERCOUNT;
ddsd.ddsCaps.dwCaps = DDSCAPS_PRIMARY-
        SURFACE |
    DDSCAPS_FLIP | DDSCAPS_COMPLEX;
ddsd.dwBackBufferCount = 1;
```

El siguiente paso será crear físicamente el primario y su *backbuffer*:

```
/Creamos el buffer primario
ddrval = lpDD->CreateSurface(&ddsd,
        &lpDDSPrimary, NULL);
```

Para crear el *backbuffer* utilizaremos el método *GetAttachedSurface* del primario creado anteriormente:

```
ddscaps.dwCaps = DDSCAPS_BACKBUFFER;
ddrval = lpDDSPrimary->GetAttachedSurfa-
        ce(&ddcaps, &lpDDSBack);
if(ddrval == DD_OK) {
// lpDDSBack apunta al backbuffer
}
else { return FALSE; }
```

- **Fase 5: renderizando los buffers**

Después de que el *buffer* primario y el *backbuffer* se han creado, renderizamos unos textos sobre ambos *buffers* utilizando funciones estándar del *GDI* de *Windows*, tal y como se muestra en el siguiente extracto de listado:

```
if (lpDDSPrimary->GetDC(&hdc) == DD_OK)
        {
    SetBkColor(hdc, RGB(0, 0, 255));
    SetTextColor(hdc, RGB(255, 255, 0));
    TextOut(hdc, 0, 0, szFrontMsg,
        lstrlen(szFrontMsg));
    lpDDSPrimary->ReleaseDC(hdc);
}
if (lpDDSBack->GetDC(&hdc) == DD_OK) {
    SetBkColor(hdc, RGB(0, 0, 255));
    SetTextColor(hdc, RGB(255, 255, 0));
    TextOut(hdc, 0, 0, szBackMsg,
        lstrlen(szBackMsg));
    lpDDSBack->ReleaseDC(hdc);
}
```

El ejemplo utiliza el método *GetDC* para obtener el *handle* del contexto del dispositivo, y éste internamente protege el *buffer*. Si no utilizamos funciones de *Windows* que requieran un *handle* del contexto del dispositivo, podemos usar los métodos denominados *Lock* y *Unlock* que son los que nos permiten proteger y desproteger el *backbuffer*.

La interfaz de DirectPlay4 hereda los métodos de DirectPlay3

- **Fase 6: escribiendo a los buffers**

La primera parte del mensaje *WM_TIMER* del ejemplo se encarga de escribir al *backbuffer*, tal y como se muestra en el siguiente extracto de código:

```
case WM_TIMER:
    // volcamos buffers.
    if(bActive)    {
        if (lpDDSBack->GetDC(&hdc) ==
        DD_OK) {
            SetBkColor(hdc, RGB(0, 0, 255));
            SetTextColor(hdc, RGB(255, 255, 0));
            if(phase) {
                TextOut(hdc, 0, 0, szFrontMsg, lstr-
        len(szFrontMsg));
                phase = 0;
            }
            else {
                TextOut(hdc, 0, 0, szBackMsg, lstr-
        len(szBackMsg));
                phase = 1;
            }
            lpDDSBack->ReleaseDC(hdc);
        }
```

La línea de código que llama al método *GetDC* prepara al *backbuffer* para su escritura. Las funciones *SetBkColor* y *SetTextColor* establecen los colores de fondo y texto. Seguidamente, la variable **phase** determina cuándo deben escribirse tanto el primario como el *backbuffer*. Si es igual a **1** se escribirá el primario y **phase** valdrá **0**. Cuando *phase* es igual a **0** se escribe en el *backbuffer* y **phase** valdrá **1**. Después de haber escrito en el *backbuffer*, éste se desprotege mediante el método *ReleaseDC*.

- **Fase 7: volcando buffers**

Para volcar el contenido del *backbuffer* al primario utilizamos el *Flip*:

```
while(1) {
    HRESULT ddrval;
    ddrval = lpDDSPrimary->Flip(NULL, 0);
    if(ddrval == DD_OK) {
        break;
    }
    if(ddrval == DDERR_SURFACELOST) {
        ddrval = lpDDSPrimary->Restore();
        if(ddrval != DD_OK) {
            break;
        }
    }
}
```

En este ejemplo, el parámetro *lpDDSPrimary* designa el primario y su *backbuffer* asociado. Cuando se ejecuta un método *Flip* los datos del primario y *backbuffer* son intercambiados (sólo los punteros de los *buffers* se cambian).

- **Fase 8: eliminando objetos DirectDraw**

Cuando pulsemos la tecla **F12** el programa procesará el mensaje *WM_DESTROY* antes de salir de la aplicación. Este mensaje llama a la función *finiObjects* que contiene todas las llamadas *Release* encargadas de eliminar los objetos *DirectDraw* antes creados :

```
static void finiObjects(void) {
    if(lpDD != NULL) {
        if(lpDDSPrimary != NULL) {
            lpDDSPrimary->Release();
            lpDDSPrimary = NULL;
        }
        lpDD->Release();
        lpDD = NULL;
    }
} // finiObjects
```

La aplicación chequea si los punteros al objeto *DirectDraw* (*lpDD*) y *DirectDrawSurface* (*lpDDSPrimary*) no son iguales a **NULL.** El programa llamará el método *Release* para decrementar el contador del objeto *DirectDrawSurface* en una unidad.

Creación de un buscador Web (y V)

Enrique de la Lastra (elastra@redestb.es)

Un *Robot Web* funciona asociado a un motor de búsqueda, de tal forma que recupera páginas *HTML* de forma automática. En este último artículo crearemos el código necesario para almacenar en una base de datos las páginas analizadas.

INTRODUCCIÓN

En este último artículo de la serie dedicada a crear un buscador *Web* dotaremos al *Robot* de la última funcionalidad imprescindible que todavía no tenía implementada: almacenar en una base de datos local toda la información que considerábamos importante sobre las páginas que se iban analizando.

El Robot cribará los enlaces antes de introducirlos en la lista

Aun así, el volcado de datos se realizará sin ningún tipo de comprobación sobre si las páginas *HTML* ya fueron indexadas anteriormente. Esta comprobación, ejecutable fácilmente desde una consulta *SQL*, será una de las posibles mejoras del *Robot*. Además se añadirán varias características y se depurarán otras, obteniendo como resultado un *Robot* plenamente funcional y ampliable de forma sencilla a las necesidades particulares de cada programador.

NUEVAS FUNCIONALIDADES

Entre las funcionalidades añadidas se realizará una criba de los enlaces antes de introducirlos en la lista de **Enlaces Analizables**. Para ello seguiremos los criterios siguientes:

1. Si un enlace hace referencia a un punto concreto de una página *HTML* en lugar de hacer referencia a la página *HTML* completa – es decir, si es un enlace relativo a un punto de la página, como por ejemplo: **inicio.htm# introduccion** – se modificará el enlace para que apunte a la página completa – **inicio.htm** en el ejemplo.
2. Si en una página *HTML* hay varios enlaces a la misma página sólo se guarda uno de ellos.
3. Si hay un enlace a un superdirectorio – como por ejemplo: "**../index .html**" -, no se tiene en cuenta, ya que, por simplicidad, se supone que la página actual se analiza gracias a un enlace situado en algún superdirectorio.
4. Por último, si hay enlaces a sitios *FTP* o a direcciones de correo no se tendrán en cuenta.

Esta limpia de enlaces requerirá además una serie de modificaciones en el código, para conseguir que no haya repeticiones en nuestra lista.

Otra mejora, para el caso del funcionamiento automático del *Robot* será la introducción de un *Timer* (temporizador) que permita lanzar la petición de la siguiente página al dispararse el evento *OnTimer.* Su razón de ser es que si no se realizan las peticiones desde el manejador de un evento, aquéllas se anidarán en el procedimiento

que las lanza, y no podremos salir de ese procedimiento hasta que no hayamos analizado todas las páginas.

Por último, se definirá una nueva variable que determine el final del análisis, y que servirá tanto para conocer el momento en el que el *Robot* termina de analizar todos los enlaces desde una página dada, como para permitir una terminación manual cuando el *Robot* trabaja de forma automática.

El análisis se podrá detener pulsando la tecla Fin del teclado

La terminación manual se realizará pulsando la tecla **Fin** del teclado; a su vez, el análisis se reactivará pulsando la tecla **Inicio.** De esta manera, podremos parar el análisis de las páginas en cualquier momento y volverlo a iniciar, en la posición en que se encontraba cuando se paró, cuando nos resulte conveniente. Esta terminación manual también puede servir para cerrar la aplicación, liberando la memoria de forma ordenada.

CRIBA DE ENLACES

Como no queremos examinar páginas previamente analizadas, sólo introduciremos en la lista aquéllos enlaces que no estén repetidos. Este cambio, requerirá un reposicionamiento del código que asignaba el nombre del servidor y la *URI* de cada enlace. Antes, este código se encontraba en el procedimiento *AnalizarSiguiente*, de forma que introducíamos todas las características de la página al llegar a dicho procedimiento.

Al indexar una página, se crean en la lista de enlaces tantas posiciones como *links* tiene la página. Si situamos

Listado 1. Procedimiento que comprueba los enlaces repetidos antes de meterlos en la lista.

```
procedure TRobotForm.EliminarEnlacesRepetidos (Enlaces: TStrings);
var
 i, j: integer;
 EnlaceExistente: string;
begin
 { Eliminamos los enlaces a superdirectorios ("../inicio.htm") y convertimos
  los enlaces relativos a una posición de la página en enlaces a la página }
 i := Enlaces.Count - 1;
 while i >= 0 do begin
  if (Pos('../', Enlaces[i]) <> 0) then
    Enlaces.Delete(i)
  else if (Pos('#', Enlaces[i]) <> 0) then
    Enlaces[i] := Copy (Enlaces[i], 1, Pos('#',Enlaces[i]) - 1);


  Dec(i);
 end;


 { Eliminamos los enlaces que estén repetidos dentro de la propia página }
 i := Enlaces.Count - 1;
 while i >= 0 do begin
   for j:= 0 to Enlaces.Count - 1 do begin
    if (i <> j) and (CompareStr (UpperCase(Enlaces[i]), UpperCase(Enlaces[j])) = 0)
    then begin
      Enlaces.Delete(i);
      break;
    end;
   end; { for }
   Dec(i);
 end; { if }


 { Eliminamos los enlaces que ya estén en la lista de enlaces }
 i := Enlaces.Count - 1;
 while i >= 0 do begin
   for j:= 0 to ListURL.Count - 1 do begin
    if pTPaginaHTML(ListURL.Items[j])^.Server <> '' then begin
      EnlaceExistente := UpperCase(pTPaginaHTML(ListURL.Items[j])^.Server +
                             pTPaginaHTML(ListURL.Items[j])^.URI);
      if CompareStr (UpperCase(Enlaces[i]), EnlaceExistente) = 0
      then begin
        Enlaces.Delete(i);
        break;
      end;
    end;
   end; { for }
   Dec(i);
 end; { if }
end; { EliminarEnlacesRepetidos }
```

el código que asigna el *URL* a las variables de la estructura de datos **PaginaHMTL** en el procedimiento *AnalizarSiguiente*, no podremos saber si están repetidos. Por tanto, situaremos este código en el procedimiento *ProcesarPaginaHTML,* justo después de conocer los enlaces válidos de una página dada.

El código necesario para detectar las referencias duplicadas se ha insertado en un nuevo procedimiento, que hemos denominado *EliminarEnlacesRepetidos*. La llamada a este procedimiento se realizará después de la del procedimiento *BuscarEnlaces,* el cual devuelve la lista de enlaces válidos:

```
{ Si se puede analizar la página, la analizamos }
if PaginaHTML^.Follow then begin
 BuscarEnlaces (Recibido,
     PaginaHTML^.Enlaces);

 if (PaginaHTML^.Enlaces.Count > 0) then
     begin
  ListBoxEnlaces.Items.AddStrings (Pagi-
     naHTML^.Enlaces);

  EliminarEnlacesRepetidos
     (PaginaHTML^.Enlaces);
```

El código del procedimiento *EliminarEnlacesRepetidos*, que comentaremos a continuación, se presenta en el Listado 1. Después de eliminar los enlaces duplicados, introducimos los restantes en nuestra lista, pero esta vez, almacenando inmediatamente el nombre del *Host*, la dirección *IP* y la *URL* de la página padre (es decir, el *Referer*):

```
  for i:= 0 to PaginaHTML^.Enlaces.Count -
     1 do
  begin
   Indice := InicializarPaginaHMTL (Pagina-
     Analizada);
   if i = 0 then
    pTPaginaHTML(ListURL.Items[Pagina-
     Analizada])^.EnlacePrimero := Indice;
   PagHTML :=
     pTPaginaHTML(ListURL.Items[Indice]);
   PagHTML^.Referer:=
       pTPaginaHTML(ListURL.Items[Pagina-
     Analizada])^.Server +
       pTPaginaHTML(ListURL.Items[Pagina-
     Analizada])^.URI;
   PagHTML^.URI := URI;
   if Servidor <> '' then begin
    PagHTML^.Server := Servidor;
    PagHTML^.Host := Servidor;
    PagHTML^.DirIP := '';
   end
   else begin
    PagHTML^.Server := DirIP;
    PagHTML^.Host := '';
    PagHTML^.DirIP := DirIP;
   end;
   ListURL.Items[Indice] := PagHTML;
  end;
 end
end;
```

En cuanto al procedimiento *EliminarEnlacesRepetidos*, éste impide la repetición de enlaces, siguiendo los criterios que se detallan a continuación: ignora los enlaces a superdirectorios (aquéllos que comienzan por **../**); convierte los enlaces relativos a un punto de una página (aquéllos que comienzan por **#**) a enlaces absolutos a esa página; elimina los enlaces duplicados, dentro de una misma página, a la misma dirección *Web*; y por último, antes de introducir los enlaces de la página en la lista de enlaces comprueba que previamente no han sido insertados.

La última de las opciones (evitar duplicados en la lista) soluciona el problema de posibles enlaces cíclicos, en los que una página enlaza con otra y ésta, a su vez, enlaza con la primera.

CONTROL DEL ANÁLISIS

El procedimiento *ProcesarPaginaHTML* termina con la llamada a un nuevo procedimiento que aglutina las funciones de actualización de las variables de control del programa y que hemos denominado *Actualizar Variables*. Este procedimiento cumplirá con las siguientes funciones:

1. Mantener actualizado el enlace padre de la página analizada.
2. Comprobar si se ha llegado a la última página de la lista.
3. Habilitar el botón que permite analizar el siguiente enlace -en el caso del funcionamiento manual- o habilitar el temporizador –funcionamiento automático-.
4. Si el análisis ha terminado, volcar la información en una base de datos para el caso en que esta opción esté habilitada.

El código completo se muestra en el Listado 2. De este listado cabe destacar el primer bloque *if,* en el que se lleva a cabo la comprobación de si se ha indexado hasta el último enlace de una página dada (página padre).

En caso afirmativo se busca la posición en la lista de la siguiente página padre (es decir, aquella página que contiene uno o más enlaces).

Al indexar una página se crean en la lista de enlaces tantas posiciones como links existen

Si al recorrer la lista no encontramos más páginas con enlaces, el índice **PaginaPadreAnalizada** alcanzará la última posición de la misma, lo cual indicará que no hay más páginas *HTML* por indexar y por tanto, que el análisis ha terminado.

Este extremo lo verificaremos mediante una variable *booleana* denominada **FinDeAnalisis.** En el Listado 2 se puede observar que sólo se realiza la llamada al procedimiento *AlmacenarBaseDatos* tras la finalización completa del análisis, y además, si la variable *booleana* **AlmacenarDatos** vale **True.**

Si queremos modificar este comportamiento, bastará cambiar de sitio la llamada a dicho procedimiento o modificar el valor de la variable denominada *booleana*.

TEMPORIZACIÓN Y PARADA

Hemos comentado que cuando el *Robot* funciona de forma automática, las peticiones *HTTP* se realizan al vencer un temporizador.

Siguiendo este mecanismo, al terminar de analizar una página se dispara un temporizador denominado **TimerSiguiente** (con una duración de 10 milisegundos). Cuando vence dicho temporizador, se recibe el evento *OnTimer* a través del cual realizamos la llamada al procedimiento *AnalizarSiguiente*, que, como recordamos, se encarga de estudiar el siguiente enlace de la lista:

```
procedure
    TRobotForm.TimerSiguienteTimer(Sender: TObject);
begin
  TimerSiguiente.Enabled := False;
  AnalizarSiguiente;
end;
```

Otra de las facilidades del *Robot* es, como también hemos comentado, la posibilidad de parar la ejecución en cualquier momento, con el fin de poder terminar la aplicación, o en el caso de que queramos seguir evaluando más tarde los enlaces restantes.

La forma de implementar este funcionamiento consiste en el cambio de unas variables *booleanas*, que a posteriori, van a detener la ejecución.

Es decir, la pausa se realiza de forma retardada, dejando a la aplicación que termine de analizar la página

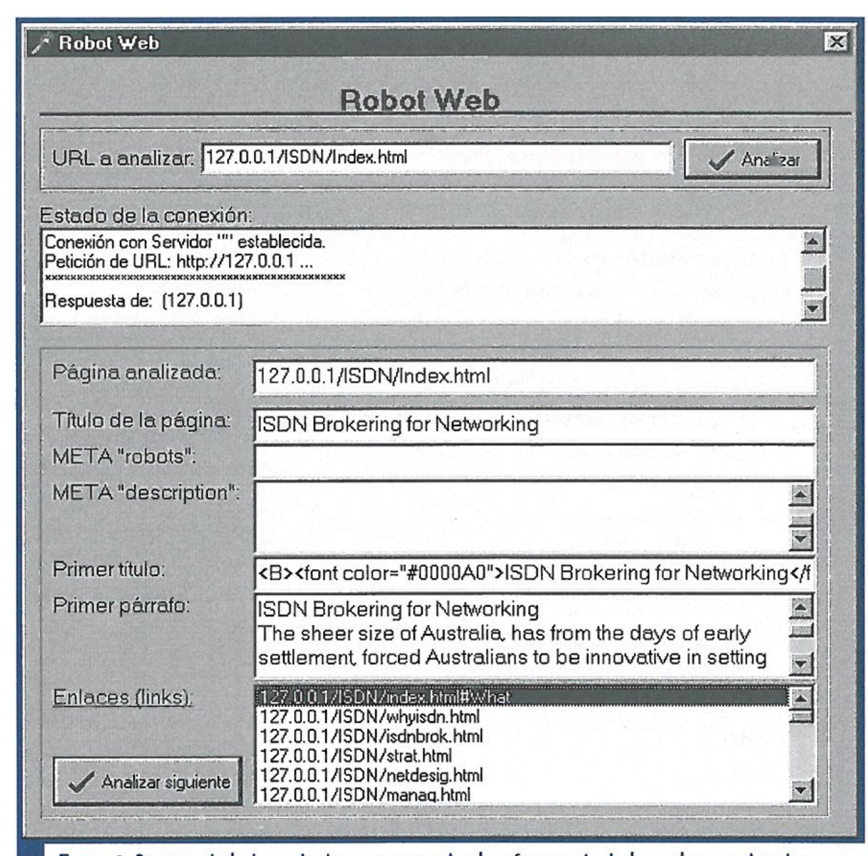


Figura 1. Campos de la base de datos que guardan la información de los enlaces indexados.

HTML sobre la que se encuentre trabajando en el momento.

El lugar donde se produce el cambio de las variables es el manejador del evento *OnKeyDown* del formulario principal. En este manejador se fijarán las variables necesarias tanto para parar la aplicación como para reanudar su marcha en el punto donde se encontraba.

El Robot permite la posibilidad de parar la ejecución en cualquier momento

Debido a que utilizamos una variable *booleana* denominada **FinDeAnalisis** para determinar el momento en que hemos llegado al último enlace de la página final, utilizaremos esta variable, en combinación con otra denominada **FinDeAnalisisParcial**, para detener la ejecución en el momento de evaluación de dichas variables:

```
procedure TRobotForm.FormKeyDown(Sender:
      TObject;
  var Key: Word; Shift: TShiftState);
begin
 if (Key = VK_END) then begin
   FinDeAnalisis := True;
   FinDeAnalisisParcial := True;
   BtnAnalizar.Enabled := True;
 end
 else if (Key = VK_HOME) and FinDeAnalisis
      then begin
   FinDeAnalisis := False;
   FinDeAnalisisParcial := False;
   BtnAnalizar.Enabled := False;
   AnalizarSiguiente;
 end;
end;
```

En cuanto a la comprobación del valor de estas variables se realiza en el procedimiento *ActualizarVariables*.

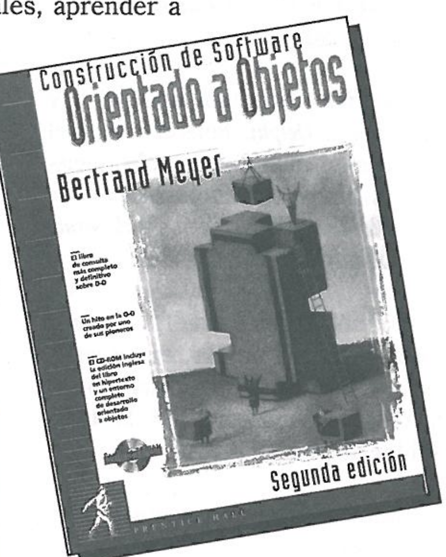


CONSTRUCCIÓN DE SOFTWARE ORIENTADO A OBJETOS

La segunda edición de este gran libro nos ofrece interesantes novedades. Algunas de éstas son por ejemplo la información incluida sobre las bases de datos orientadas a objetos, y la persistencia y evolución de esquemas. Por otra parte está el diseño por contrato, es decir, cómo construir software fiable. También se pueden encontrar datos sobre el estudio de los patrones de diseño fundamentales, aprender a encontrar las clases y muchos otros temas de metodología.

Además el libro incluye más de 400 referencias a libros, artículos, páginas web y foros de discusión y un gran glosario sobre la tecnología de objetos y sus características.

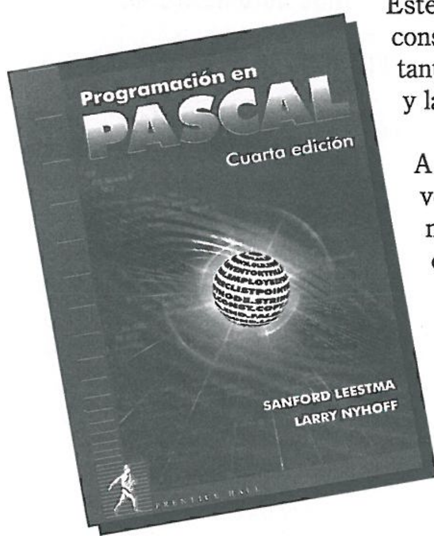
El CD-ROM adjunto contiene la edición inglesa del texto en hipertexto y un completo entorno de desarrollo orientado a objetos. Además incluye unos componentes de biblioteca que aportan un extenso material para ayudar en el estudio del apéndice.

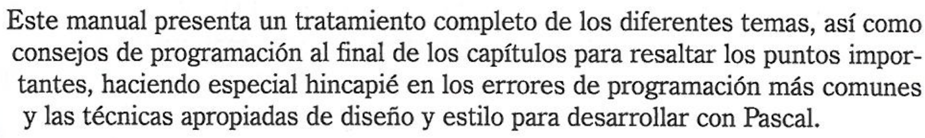


Editorial: Prentice Hall	Autor: Bretrand Meyer
Nº páginas: 1.198	Idioma: Español
Nivel: Avanzado	Precio: 8.665 Ptas. (I.V.A. inc.)

PROGRAMACIÓN EN PASCAL

El libro proporciona una base muy sólida para conocer los fundamentos de la programación en Pascal, de forma clara, concisa y completa. Esta edición, la cuarta, al igual que las anteriores mantiene las mismas características en lo referente a su componente didáctico y práctico.





Este manual presenta un tratamiento completo de los diferentes temas, así como consejos de programación al final de los capítulos para resaltar los puntos importantes, haciendo especial hincapié en los errores de programación más comunes y las técnicas apropiadas de diseño y estilo para desarrollar con Pascal.

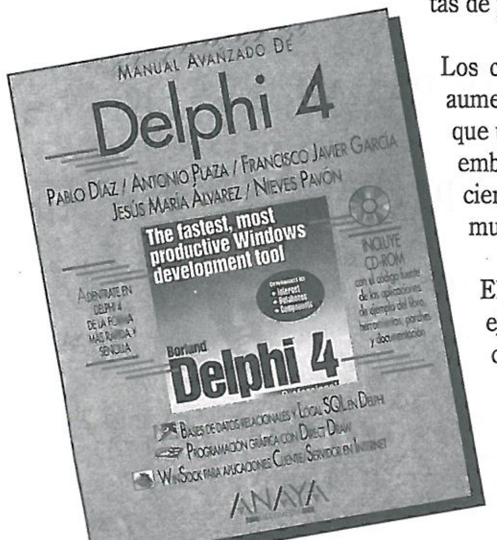
A lo largo de todo el texto se ofrecen una amplia cantidad de ejemplos que sirven para ilustrar los conceptos y la estructura a seguir, también aparecen numerosos ejercicios tanto breves como extensos y proyectos de programación de un amplio rango de áreas de aplicación.

Este volumen contiene un disquete con más de 60 programas y ejemplos completos de programación en Pascal.

Editorial: Prentice Hall	Autores: S.Leestma y L.Nyhoff
Nº páginas: 824	Idioma: Español
Nivel: Avanzado	Precio: 5.995 Ptas.(I.V.A. inc.)

MANUAL AVANZADO DE DELPHI 4

Este interesante y novedoso manual intenta dar una visión global de las principales tecnologías existentes en el mundo de la programación. Sabido es por todos la dificultad que supone el abordar todos los contenidos, pero es posible intentar mostrar las principales ramas de los mismos, todo ello haciendo uso de una de las herramientas de programación más potentes del mercado, hablamos por supuesto de Delphi 4.



Los compiladores aumentan en prestaciones y en facilidad de uso, pero también aumentan en complejidad, habitualmente podríamos pensar que Delphi 4 no es más que un compilador tradicional que ha adaptado sus ventanas y menús a la moda, sin embargo, detrás de este tipo de aplicación, existe un motor mucho más potente y cientos de nuevas librerías que nos permiten realizar el trabajo de una forma mucho más sencilla y eficiente.

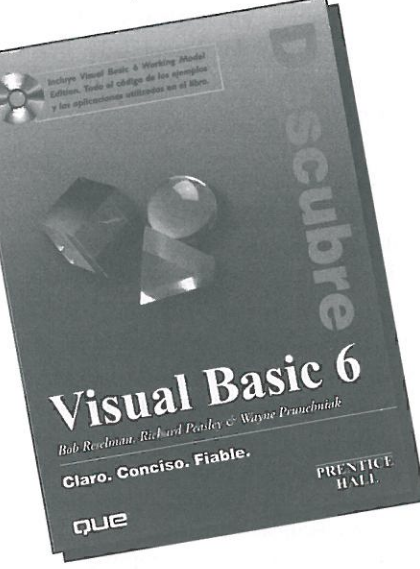
El libro además incluye un CD-ROM con el código fuente de las aplicaciones de ejemplo aparecidas a lo largo del manual, así como también herramientas, parches y una amplia documentación.

Editorial: Anaya Multimedia	Autores: P.Díaz, A.Plaza, F.García, J.Álvarez y N.Pavón
Nº Páginas: 464	Idioma: Español
Nivel: Avanzado	Precio: 3.295 Ptas.(I.V.A. inc.)

DESCUBRE VISUAL BASIC 6.0

De nuevo esta importante editorial nos sorprende con un interesante y completo manual sobre la novedosa versión 6 del archiconocido lenguaje de programación Visual Basic.

Un libro que no sólo está orientado a los programadores principiantes sino que también aquellos desarrolladores con experiencia, y que están más familiarizados con otros lenguajes, encontrarán en este volumen todo lo que desean aprender. El principal objetivo que pretende cumplir el manual es el de enseñar a programar mostrando los diferentes programas paso a paso o bien desde un punto de vista conceptual.



En este volumen se incluyen todas las nuevas características que incorpora Visual Basic 6.0. Aprenderás a comprender el funcionamiento de eventos, métodos, diálogos y gráficos, a crear controles ActiveX, aplicaciones de bases de datos e informes.

El libro está acompañado por un CD-ROM que incorpora una Working Model del programa así como todo el código de los ejemplos y las aplicaciones utilizadas en las explicaciones expuestas en el manual.

Editorial: Prentice Hall	Autores: B.Reselman, R.Peasley y W.Prunchniak
Nº páginas: 676	Idioma: Español
Nivel: Principiante-intermedio	Precio: 5.700 Ptas.(I.V.A. inc.)

Dudas técnicas

En esta sección, los lectores de SÓLO PROGRAMADORES tienen la oportunidad de hallar respuesta a los problemas que puedan tener en algún tema relacionado con la programación en cualquier entorno.

Pregunta

Hola amigos:

Compro vuestra revista habitualmente y la encuentro muy interesante. Hace algunos días he instalado *Linux* en mi ordenador, un *Pentium II* 400 Mhz, con el fin de realizar mis pequeños pinitos programando en este sistema.Tengo bastante experiencia programando en *MS-Windows* y *Windows 95* con *Visual Basic*, *Visual C++* y *Delphi*, sin embargo no consigo realizar aplicaciones complejas en *Linux*. Normalmente programo en lenguaje *C*, por lo que estoy utilizando el compilador que viene instalado en la distribución que se llama *GCC*.

Si creo un único fichero y lo compilo, no hay ningún problema, sin embargo no encuentro ninguna manera de crear un proyecto o algo similar en *Linux*. Al no haber un entorno de desarrollo, no sé donde buscar la información. He buscado en el *Emacs* y no hay manera de crear un proyecto en ninguno de los menús. ¿Podríais ayudarme a crear uno? Además, aunque conozco las llamadas estándar de *C*, las específicas de *Linux* no se cuales son, así que me encuentro algo perdido. Ya sé que con el comando man puedo encontrar información sobre una función, pero para eso tengo que sabérmelas. ¿No hay ningún sistema para acceder a la lista completa? Muchas gracias por adelantado.

Respuesta

Estimado Lector:

Ante todo felicitarte por tu elección. La programación en un sistema *Linux* es muy interesante, ya que al ser un entorno de 32 bits, tienes a tu alcance toda la potencia del microprocesador. Además, es un primer paso para adentrarte dentro del mundo *Unix*, que tanta difusión tiene en los entornos corporativos de las grandes empresas.El *Emacs* es increíblemente extensible, pudiendo crear tus propios modos, incluidos métodos para dictar a través de la tarjeta de sonido del PC. Además, puedes acceder a las páginas de manual directamente desde tu código: basta con marcar la función que quieras conocer y activar el comando *Alt-x manual-entry <enter> <enter>*. Sin embargo, si echas de menos los iconos y el sistema de *Windows*, vamos a sugerirte dos alternativas:

1) La primera de ellas es el *Xemacs*, una variante del emacs normal, que incorpora una serie de mejoras en el apartado del interfaz de usuario. Este editor tiene la ventaja de manejarse exactamente igual que el emacs, al que añade iconos y menús que facilitan la utilización del mismo.
2) La segunda es *Xcoral*. No es exactamente igual que emacs, aunque comparten una gran cantidad de comandos, por lo que la transición de uno a otro es muy sencilla. Además, este editor posee un navegador de funciones y estructuras que hace enormemente cómoda la lectura del código.

A la hora de programar en un entorno, es imprescindible conocer las diferentes llamadas que te permite realizar. Como tu bien dices, con el comando man, es posible conocer los detalles de una función que ya usamos. Si quisiéramos saber que hace la primitiva *select*, basta con teclear en una terminal:

```
man select <enter>
```

Sin embargo, a veces no conocemos que recursos nos ofrece el sistema, y no tenemos a mano un libro que nos ofrezca la solución. En este caso, es imprescindible un compendio de las

diferentes llamadas al sistema, así como de las distintas funciones de librería disponibles.

Una posible opción, sería la de utilizar la opción -K que nos ofrece el programa man. Invocado con ella el sistema busca la palabra que le insertemos a continuación, sin necesidad de que sea el nombre de una función, en todas las páginas de manual disponibles. En el ejemplo anterior, podríamos hacer lo siguiente:

```
man -K select
```

Aunque interesante, esta opción es bastante lenta, ya que busca todas las ocurrencias en el interior de todos los ficheros disponibles. Si lo que deseamos es simplemente una lista en la que nosotros podamos consultar las diferentes llamadas, te recomendamos *Xman*. Con esta herramienta podrás acceder a todas las páginas instaladas y consultarlas con un sólo toque de ratón.

Finalmente llegamos al último punto que es, probablemente, el más importante.

En el mundo *Unix* no existe el concepto de proyecto, sino que se utiliza la herramienta *Make*. Esta utilidad permite definir una serie de parámetros o reglas de ejecución que nos van a permitir realizar, entre otras cosas, compilaciones de grandes proyectos sin necesidad de realizar todas las operaciones manualmente.

Sin embargo, como todas las herramientas existentes en *Unix*, su utilidad no se limita a la generación de ejecutables, sino que es posible utilizarla para muchos otros fines.

Para emplearla daremos los siguientes pasos:

1) Generamos un fichero llamado *Makefile*, en el cual escribiremos todas las reglas que tenga que cumplir nuestro conjunto de ficheros fuente.
2) Una vez generado el fichero, podremos compilar nuestro programa utilizando la orden *make*.
3) Si el fichero creado no se llama *Makefile*, podremos compilar tecleando *make -f* <fichero>.
4) Si hemos definido varios puntos de entrada en el *Makefile*, podremos acceder directamente a uno de ellos con *make* <punto de entrada>.

A pesar de que existe una gran cantidad de documentación al respecto, la construcción de *Makefiles*, es un tanto compleja dada su potencia y versatilidad. Para facilitar las cosas, vamos a tratar de construir un pequeño ejemplo en el que se vea el modo de empleo.

Para nuestro ejemplo vamos a suponer que tenemos un pequeño proyecto que se compone de cuatro ficheros, todos los cuales están en el mismo directorio:

main.c
logo.c
final.c
programa.c

Además, nuestro programa ha de enlazarse con las siguientes librerías, presentes en */usr/lib*:

libtermcap
libncurses

Todo *Makefile* posee dos secciones:

1) La primera de ellas contiene las declaraciones de las diferentes variables disponibles. Siempre vamos a pasar por esta sección, independientemente del punto de entrada.
2) La segunda define todos los pasos a dar, cada uno de los cuales contendrá uno de los punto de entrada.

Vamos a crear el fichero poco a poco. Comenzaremos definiendo el compilador y sus *flags* de optimización.

```
CC=gcc
CFLAGS= -O6 -funroll-loops
OPTFLAGS= -Wall -pedantic
```

A continuación creamos una lista de directorios donde el compilador ha de buscar las cabeceras y las propias librerías:

```
INCPATH= -I/usr/include
LIBS= -lncurses -ltermcap
LIBPATH=-L/usr/lib
```

Luego definimos los ficheros fuente y objeto, así como el ejecutable.

```
SRCS=main.c logo.c programa.c final.c
OBJS= main.o logo.o programa.o final.o
EXEC=ejemplo
```

Acabadas las definiciones, vamos a definir los puntos de salto. En ellos nos referiremos a las diferentes variables como *$*(variable). El primero de ellos es al que se accede por defecto si no indicamos lo contrario en la invocación del comando *make*.

El primer punto de salto, llamado all se completará al ejecutar, en el orden indicado, los demás puntos correspondientes a sus dependencias.

```
all: compile link
```

A continuación definimos la entrada compile, en la cual vamos a crear los códigos objeto a partir de los fuentes de nuestro programa.

```
compile: $(SRCS)
        $(CC) $(CFLAGS) $(OPT-
FLAGS) -c $< $(INCPATH)
```

Después, generaremos el ejecutable enlazando todos los ficheros objeto.

```
link:   $(OBJS)
 $(CC) $(CFLAGS) $(OPTFLAGS) -o
$(EXEC) $(LIBPATH) (LIBS)
```

Finalmente, creamos una última entrada, que no se ejecutará a no ser que hagamos *make clean*, en la que borramos todos los ficheros que no nos sean útiles para dejar un árbol de fuentes limpio.

```
clean:
  -rm $(OBJS)
  -rm $(EXEC)
  -rm *~
  -rm core
```

Con este fichero hemos creado cuatro entradas, cada una de las cuales hace lo siguiente:

All: genera el ejecutable a partir de los fuentes.
Compile: genera los .o a partir de los fuentes.
Link: genera el ejecutable a partir de los .o.
Clean: limpia el directorio.

El fichero final queda de la siguiente manera:

```
#
# Makefile ejemplo. Las líneas que
comienzan por '#' son comentarios.
#


#######################
#######################
#######
#
# Definiciones

CC=gcc
CFLAGS= -O6 -funroll-loops
OPTFLAGS= -Wall -pedantic
INCPATH= -I/usr/include
LIBS= -lncurses -ltermcap
LIBPATH=-L/usr/lib
SRCS=main.c logo.c programa.c final.c
OBJS= main.o logo.o programa.o final.o
EXEC=ejemplo


#######################
#######################
#######
#
# Puntos de entrada

all: compile link
compile: $(SRCS)
        $(CC) $(CFLAGS) $(OPT-
FLAGS) -c $< $(INCPATH)
link:   $(OBJS)
  $(CC) $(CFLAGS) $(OPTFLAGS) -o
$(EXEC) $(LIBPATH) (LIBS)
clean:
  -rm $(OBJS)
  -rm $(EXEC)
  -rm *~
  -rm core
```

Pregunta

Hace tiempo que leo vuestra revista y veo con mucho interés los artículos que publicáis sobre tarjetas de vídeo, tanto las que son *3D* como las que no.

En concreto me han parecido muy interesantes artículos como el que apareció en su día de como realizar un driver de tarjeta de vídeo o bien el más reciente dedicado a las tarjetas *3Dfx*.

Yo mismo estoy intentando hacer mis pinitos con algunas tarjetas antiguas que tengo, utilizando para ello un 486 que ya no se usa para nada en mi casa. Hasta ahora he estado documentándome y creo que ya tengo todo lo necesario para realizar los programas que quiero, excepto por una cosa.

He visto en el excelente conjunto de utilidades *VGADOC4b*, un montón de referencias relativas a los modos de acceso de las tarjetas de vídeo. En concreto, siempre habla de dos métodos:

MMIO.
Método VGA estándar.

Si es posible, me gustaría que me explicárais las diferencias entre ambos ya que no se cual elegir.

Muchas gracias por adelantado.

Respuesta

Como muy bien sabes, el estándar que *IBM* definió para el PC, contempla dos tipos de espacios de direcciones:

Espacio de memoria.
Puertos de E/S (Entrada y Salida).

El método tradicional de las tarjetas de vídeo es acceder a los registros internos a través de unos puertos de entrada y salida determinados, mientras que los datos de imagen se colocan directamente en la apertura que la *BIOS* reserva para la memoria de vídeo.

Normalmente, el acceso a los puertos de entrada salida es mucho más lento que los accesos normales a memoria. Sin embargo, en las tarjetas originales no era muy relevante porque los accesos de control eran muy pocos en proporción con los accesos de datos y los buses ISA no contribuían precisamente a diferenciar ambos tipos.

Con la llegada de *VLB* y más tarde *PCI*, el tiempo que se tarda en trasvasar los datos es cada vez menor, mientras que la velocidad de acceso a los puertos de entrada/salida permanece constante.

Además, la aparición de aceleradores cada vez más complejos que necesitan de mayor cantidad de instrucciones disminuyendo la cantidad de datos que atraviesan el *bus*, hace que la cantidad de tiempo que la tarjeta y el micro están accediendo a los puertos sea demasiado grande. Debido a ello aparece lo que se llamó *Memory Mapped IO*, o lo que es lo mismo, entrada y salida a través de la memoria. De esta manera, se combinan los flujos de datos y control en un único interfaz, evitando la latencia de los puertos.

Por lo tanto, siempre que podamos utilizaremos *MMIO* sobre *I/O* convencional, ya que la velocidad de la primera es mucho mayor que la segunda.

No haga esperar a sus Clientes. Con DB2 Universal Database, todos podrán ser el primero de la fila.

Porque DB2 Universal Database es la base de datos diseñada para hacer negocios a través de Internet.

Completamente compatible con Java, proporciona soporte para textos, gráficos, sonido y vídeo para poner en marcha aplicaciones Self-Service en Internet.

Y todo ello simultáneamente y al instante para miles de usuarios, de modo que todos sean los primeros y reciban la mejor atención sin tener que esperar colas.

Además, sus completas funciones de conectividad facilitan la consolidación de datos provenientes de cualquier fuente y en cualquier plataforma. Y, a propósito de plataformas, DB2 Universal Database funciona en modo nativo en una gran cantidad de ellas, desde Windows NT hasta Sun Solaris y AIX.

Visítenos en www.software.ibm.com/webservers/websoftware/sp


e-business

Consiga sin cargo alguno un kit de iniciación del software Web Self-Service de IBM que incluye un CD de demostración de DB2 Universal Database en www.software.ibm.com/webservers/websoftware/sp



Soluciones para nuestro pequeño mundo